ABC INFORMATIQUE

N° 15

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Voici les robots

Aquarius

Des souris et des hommes

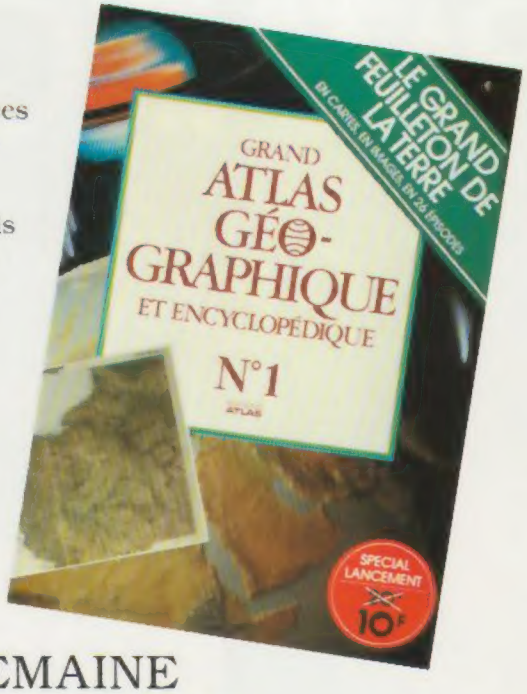
L'arbre des programmes

EDITIONS ATLAS

M6062-15-12F 85FB-3,80FS-$1.95

GRAND ATLAS GÉO-GRAPHIQUE ET ENCYCLOPÉDIQUE

Voici la somme des connaissances actuelles de la science cartographique et géographique.
Une équipe de savants et de géographes internationaux a mis au point cet Atlas exceptionnel, avec un souci de précision et de clarté qui en fait un document aussi irremplaçable que passionnant.
En 26 épisodes, vous reconstituez vous-même l'intégralité de ce très bel ouvrage.
Le n° 1 est en vente dans un luxueux dossier chez tous les marchands de journaux au prix de lancement de 10 F (75 FB) seulement.



SEMAINE APRÈS SEMAINE, POUR UN PRIX MODIQUE, CONSTITUEZ-VOUS UN FABULEUX ATLAS.

Dès le premier numéro, vous découvrirez l'extraordinaire richesse de la cartographie. Un guide de lecture et un index vous aideront à y découvrir une mine de renseignements. Lorsque votre collection sera complète, vous la conserverez, protégée par deux luxueuses reliures, en bonne place dans votre bibliothèque familiale.

EDITIONS ATLAS

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles, permettant de relier 12 fascicules, seront en vente en permanence chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume II, n° 15.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, J.-P. Bourcier *(coordination)*, S.I.-André Larochelle *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Photo Matra.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : avril 1984. 13844. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Votre dévoué serviteur

Les robots industriels peuvent maintenant reconnaître la forme des objets et remplacer l'homme dans de nombreuses tâches.

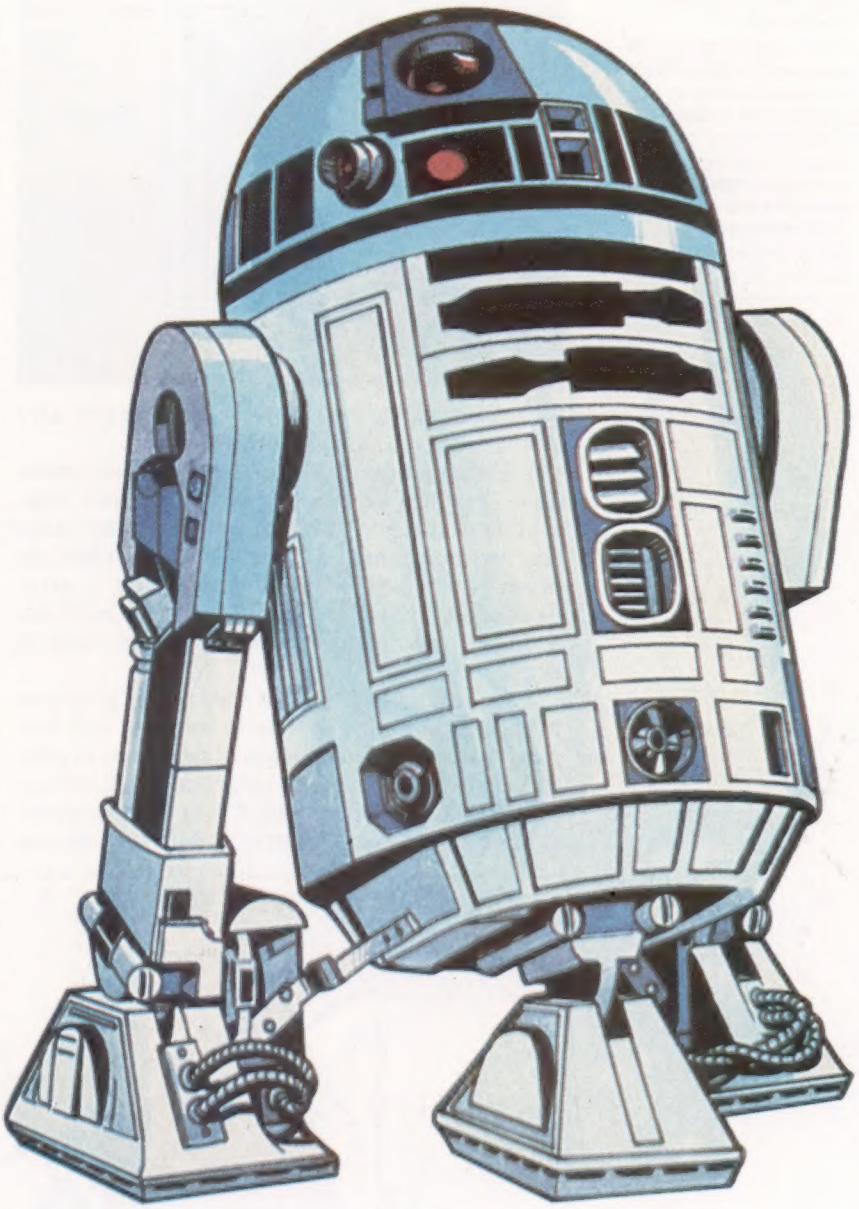
Le terme « robot » vient du tchèque *robota,* qui signifie travail. Ce mot fut inventé par l'écrivain Karel Čapek dans sa pièce *R.U.R. (Rossum's Universal Robots),* écrite en 1920, et fut par la suite adopté par les écrivains de science-fiction. Malgré l'image de puissance que leur donnent les romans de science-fiction, les robots ne sont rien d'autre qu'une extension mécanique de l'ordinateur, avec autant de restrictions et de défauts.

Leurs origines se situent dans les ateliers des années cinquante, au moment où la mise au point des commandes numériques pour machines-outils fut élaborée. Ces premiers efforts étaient, comme on peut l'imaginer, plutôt sommaires : des machines étaient commandées à partir de bandes de papier perforé et pouvaient au mieux déplacer un outil fixe d'un point à l'autre sur l'objet usiné.

L'étape suivante de leur développement fut l'introduction de la possibilité de changer d'outil au cours d'un travail. Cela était faisable grâce à la présence d'un « carrousel », ou support rotatif d'outils; ces outils possédaient tous des points d'ancrage identiques et pouvaient être sélectionnés et fixés au porte-outil placé sous la commande d'un programme.

Malgré ce perfectionnement, une machine particulière ne pouvait effectuer plus d'une tâche : un tour était toujours un tour, même s'il pouvait effectuer automatiquement toutes les opérations de tournage impliquées dans un processus particulier. Presque au même moment, des mains et des bras mécaniques commandés à distance furent créés pour travailler dans des conditions dangereuses, par exemple sous la mer ou lors de manipulations de matériaux radioactifs. Ces appareils de manipulation n'étaient simplement que des extensions des propres mains de l'opérateur; mais on utilisa très rapidement les ordinateurs pour les commander directement. Les robots qui ont été développés depuis lors sont plus précisément nommés « bras articulés », puisqu'ils consistent en un porte-outil monté sur un bras articulé.

Pour comprendre comment les robots sont programmés, nous devons d'abord les situer par rapport à leur espace de fonctionnement. La plupart des robots industriels sont fixés sur place. L'espace de travail est donc une sphère dont la partie inférieure est plate. Nous pouvons donc réduire les problèmes de commande du robot à un simple exercice de géométrie tridimensionnelle. Le centre de la sphère se situe au niveau de l'articulation de l'« épaule » du robot, et le rayon est la longueur du bras, mesurée de l'« épaule » au bout des « doigts » — la pince ou le porte-outil. Tout point situé à l'intérieur de cet espace peut être exprimé en trois coordonnées : par exemple, en termes de distance vers le nord, le sud, l'est ou l'ouest et vers le bas ou le haut, à partir d'un point zéro. Dans ce cas, les coordonnées sont dites « cartésiennes ». La position peut également être exprimée en coordonnées « sphériques ». Cela serait traduit dans un langage de tous les jours : une distance de 2 m dans une direction nord-est et située à



Héros mécanique
R2D2, le robot attachant de *La Guerre des étoiles,* était en fait commandé par un opérateur humain. Sa conception reflète l'image que de nombreuses personnes se font d'un robot.

Robot alimenté par piles
Le Hero-1 est un robot entièrement autonome, alimenté par piles et qui présente certaines des fonctions d'une tortue ainsi que les possibilités de manipulation d'un bras robot. Coûtant environ 27 000 F — ou 19 000 F en kit — il peut sembler être un jouet de luxe. Mais en fait c'est un système informatique remarquablement souple, avec des caractéristiques évoluées comme la synthèse de la parole, la photodétection, une entrée auditive et (puisqu'il est mobile) un système de détection des distances à ultrasons qui peut également servir de système de détection de mouvement.
(Cl. Ian McKinnell.)

30° au-dessus de l'horizontale. La position zéro dans ce cas est l'« épaule » du robot.

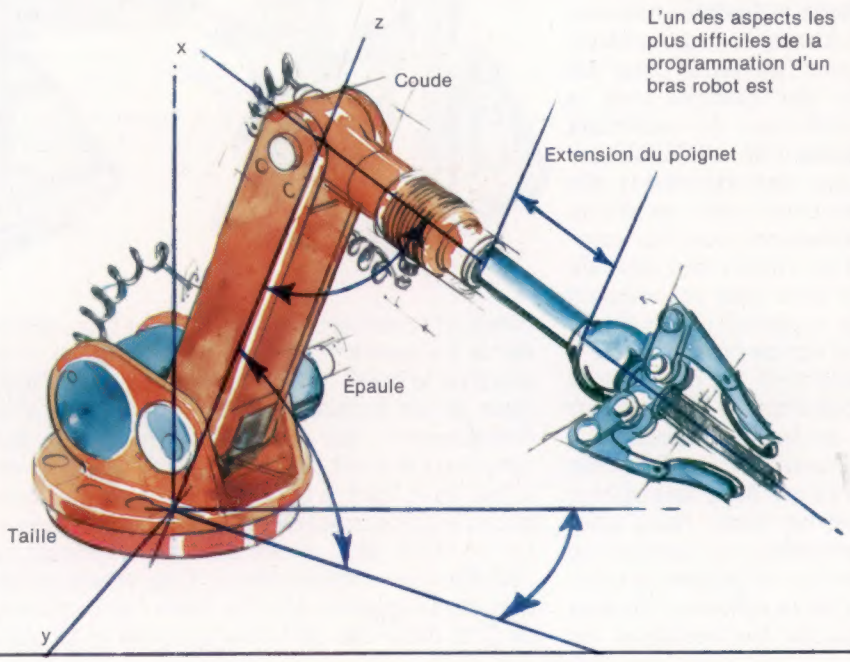
La programmation du robot implique cependant l'envoi d'un ensemble d'instructions commandant divers déplacements. Il existe donc une troisième méthode de positionnement du porte-outil; connue sous le nom de « positionnement point par point », cette méthode nécessite que le point zéro se déplace avec le porte-outil.

Les robots industriels ont une précision moyenne de 1 mm. Même les modèles très simples, vendus quelques milliers de francs et pouvant être utilisés par tout ordinateur domestique muni d'une sortie parallèle 8 bits, ont une précision de 2 mm. Cette observation est intéressante puisque la différence de coût est de 50 pour 1. Deux méthodes sont généralement adoptées pour commander des bras articulés. Pour ceux qui manipulent des objets de faible poids, des moteurs pas à pas (moteurs électriques dont les mouvements prédéterminés sont sollicités chaque fois qu'un courant leur est appliqué, comme ceux qui sont utilisés dans les lecteurs de disquettes pour positionner la tête de lecture-écriture) sont suffisants. Mais, pour les robots utilisés dans les chaînes de montage, où des charges lourdes doivent être manœuvrées, on utilise plus fréquemment des béliers hydrauliques pour bouger les diverses parties du bras autour de leurs pivots (leurs points d'appui, d'articulation). Il est assez simple de mesurer le volume de fluide hydraulique qui est injecté dans les bras, et de déduire le mouvement produit pour obtenir ainsi une précision de positionnement acceptable.

Les robots industriels renferment toujours un mini-ordinateur intégré fait sur mesure (ou un micro-ordinateur à grande capacité sur les modèles plus récents) dont la fonction se limite à commander le bras et à exécuter un langage de programmation conçu à cet effet. Comme ce langage ne doit qu'indiquer des coordonnées et émettre des commandes simples comme FERMER PINCE ou OUVRIR PINCE, le langage de programmation ne possède aucune instruction pour manipuler du texte. Les instructions du programme sont entrées à l'aide d'un clavier numérique relié à l'ordinateur au moyen d'un long « cordon ombilical ». Cela permet à l'opérateur de se déplacer autour du robot tout en entrant des instructions. Les versions les plus évoluées de ces dispositifs de commande comportent une manette très précise.

Il existe une autre méthode de programmation employée surtout dans les situations où il

Mouvement angulaire



L'un des aspects les plus difficiles de la programmation d'un bras robot est la conversion géométrique. Nous avons l'habitude de spécifier des positions à l'aide de coordonnées x, y, z. Il est nécessaire de communiquer au robot des valeurs d'angles pour les articulations du « coude » et de l'« épaule », un degré de rotation pour la « taille », et une distance d'extension du « poignet ». Dans les systèmes bas de gamme, les programmeurs doivent fournir ces quatre valeurs. Des robots plus sophistiqués peuvent effectuer toutes les conversions à partir de coordonnées cartésiennes.
(Cl. Bob Freeman.)

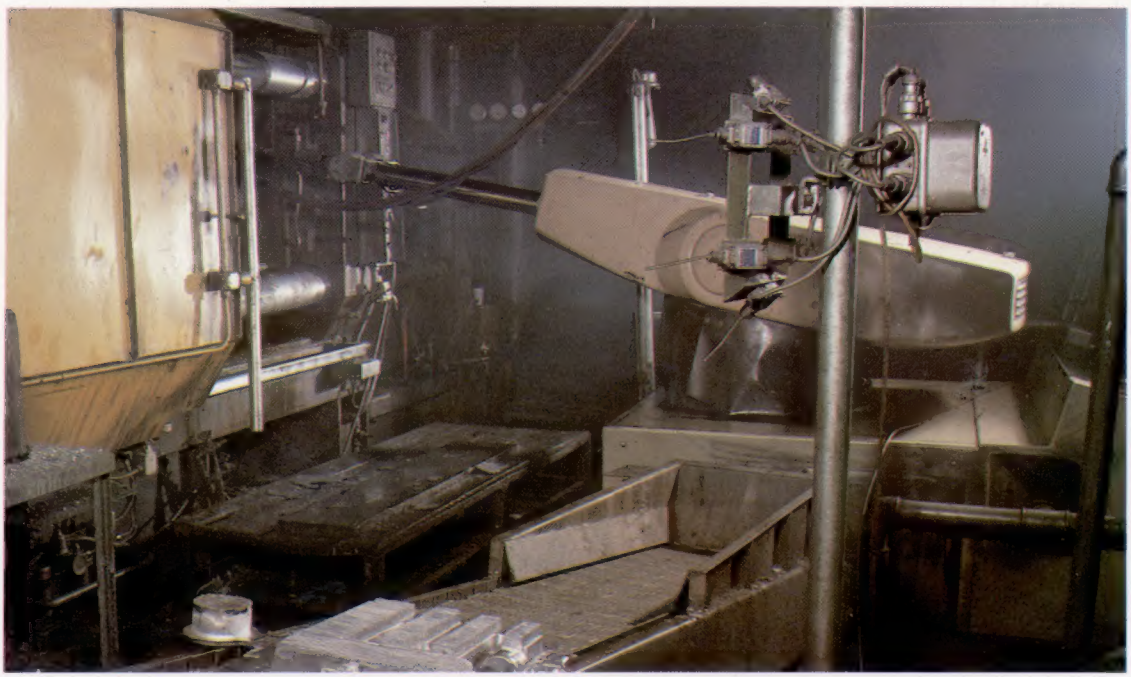
n'est pas obligatoire de positionner les outils de façon très précise, comme dans des travaux de peinture au pistolet. Ici, l'opérateur déplace le bras du robot afin de lui indiquer les divers mouvements nécessaires pour effectuer un travail et entre ainsi directement dans la mémoire de l'ordinateur les diverses interventions. Le robot n'aura alors qu'à répéter ces mouvements lors de chaque exécution du programme.

Dans toutes ces méthodes, il n'est nécessaire de définir que la position du porte-outil. L'opérateur n'a pas à se soucier des positions relatives des sections individuelles du robot. Celles-ci sont définies par le langage de programmation résidant dans l'ordinateur qui commande le robot et calcule toutes les optimisations utiles.

Les robots dont nous avons parlé jusqu'ici ne sont capables que d'obéir aveuglément en répétant la même tâche, exactement au même endroit, quelles que soient les influences externes. On les utilise principalement dans l'industrie d'équipement, spécialement dans la construction automobile. Ils sont disposés en chaîne de montage, où la pièce ou le véhicule partiellement terminé est toujours précisément localisé dans le temps et dans l'espace. Cela est particulièrement important pour mener à bien un processus de production par robot, puisque, si la pièce est mal positionnée, le robot ne pourra adapter son intervention. Pour tenter de résoudre ce problème, divers capteurs peuvent être montés sur le porte-outil. Le plus simple de ces dispositifs pourrait être un micro-interrupteur. Des procédures d'urgence pourraient être intégrées dans le programme de commande (une commande ATTENDRE, par exemple), celles-ci étant exécutées si le micro-interrupteur n'entre pas en contact avec la pièce à manipuler, mais des procédures plus sophistiquées nécessiteraient l'intervention humaine.

Il serait possible de remplacer cette détection par pression en utilisant un photodétecteur. Un photodétecteur placé sur le porte-outil pourrait repérer la présence de la pièce lorsque celle-ci voilerait une source lumineuse. Le porte-outil pourrait ainsi être arrêté avant de heurter la pièce, mis en attente jusqu'à ce que la pièce soit en position, et ensuite autorisé à poursuivre. Évidemment, des erreurs sont toujours possibles ; dans les situations exigeant une grande fiabilité, il est possible d'installer un système de reconnaissance d'images utilisant des caméras à couplage de charge (CCD). Ces caméras transmettent directement l'image sur une puce à traitement matriciel (une puce fractionnée en plusieurs centaines de photodétecteurs, dont chacun est en mesure d'enregistrer non seulement le noir et le blanc, mais aussi toute une gamme de tons intermédiaires). Chaque photodétecteur individuel peut nécessiter 1 octet de mémoire pour définir le contraste dans l'échelle de gris. Au départ, chaque objet est « photographié » de nombreuses fois, et un programme d'apprentissage calcule une moyenne des valeurs. Lors de l'exécution, la caméra CCD transmet l'image de l'objet, qui est comparée à l'image de référence stockée en mémoire. Si ces deux images correspondent, l'opération peut se poursuivre. Il est ainsi possible de vérifier la présence de la pièce nécessaire, ainsi que l'exactitude de sa position.

Ce système de traitement d'image sert aussi à sélectionner et à positionner des composants posés en vrac. Cette application devient de plus en plus fréquente pour de petits robots et fait de ceux-ci des accessoires auxiliaires dans la chaîne de montage. Outre leur utilisation dans le processus de production lui-même, les robots industriels sont souvent employés dans les opérations de vérification et de contrôle.



Au travail
Les bras robots, comme celui-ci photographié au travail dans un atelier de moulage, se chargent de plus en plus des travaux les plus pénibles et les plus dangereux effectués en usine. Le nettoyage des pièces moulées avant leur usinage est un bon exemple. L'objet venant tout juste d'être moulé est encore beaucoup trop chaud pour être manipulé par des mains humaines, et devrait normalement reposer quelques minutes afin de permettre son refroidissement. Le robot n'est pas sensible à la chaleur et peut manipuler la pièce immédiatement et lui faire subir sans délai la prochaine opération.

Fonctions « son » du Vic

Examinons de près la production de sons sur le Vic-20...

Le Vic-20 fut un des premiers ordinateurs domestiques qui apparurent en France. Ses possibilités peuvent donc maintenant sembler désuètes en comparaison de celles d'ordinateurs plus récents. De plus, Commodore n'a rien fait pour simplifier l'écriture des programmes musicaux ou sonores puisque le BASIC du Vic-20, comme celui du Commodore 64, n'offre aucune commande associée spécifiquement au son. Toute la commande sonore est définie par une série d'instructions POKE dirigées vers divers emplacements mémoire. Ce principe s'applique également au Commodore 64, et les techniques décrites ici seront aussi très pratiques. Les commandes sonores disponibles se limitent au volume (ce qui signifie que les composants d'enveloppe, d'attaque, de chute et de fin égalent zéro) et au réglage de fréquences sur trois oscillateurs et sur un générateur de bruit. La sortie ne peut être produite que par un haut-parleur de téléviseur. De plus, en raison de l'imprécision avec laquelle le Vic-20 sélectionne les fréquences, il est impossible de reproduire exactement toutes les notes de la gamme.

Avec ce potentiel limité, le Vic-20 n'a que peu de valeur sur le plan musical.

Commande sonore

Le Vic-20 possède trois oscillateurs d'onde carrée et un générateur de bruit. Chaque oscillateur couvre approximativement trois octaves.

Osc. 1	Osc. 2	Osc. 3	Gamme de fréq. (Hz)	Octave
•			(65,41-123,47)	1
•	•		(130,81-246,94)	2
•	•	•	(261,63-493,88)	3
	•	•	(523,25-987,77)	4
		•	(1046,5-1975,53)	5

Cette disposition permet à l'utilisateur de couvrir un total de cinq octaves, tout en disposant d'au moins un oscillateur dans chaque octave. L'octave 3 qui débute au *do* sous le *la* du diapason et qui renferme le *la* du diapason à 440 Hz est accessible à l'aide des trois oscillateurs.

La commande des oscillateurs est effectuée en modifiant ainsi le contenu de cinq adresses :

Adresse de mémoire	Oscillateur
POKE 36874,X	1
POKE 36875,X	2
POKE 36876,X	3
POKE 36877,X	bruit

Dans chaque cas, X est un nombre entier compris entre 135 et 241 (0 met cet oscillateur hors fonction), qui correspond à des valeurs dont les notes équivalentes sont données à la page 73 de la notice qui accompagne le Vic-20. Afin de pouvoir entendre la fréquence sélectionnée, le volume doit être réglé de la façon suivante :

```
POKE 36878,V
```

où V peut être choisi entre 0 (aucun son) et 15 (fort) ce qui règle tous les oscillateurs et le générateur de bruit. Par exemple :

```
POKE 36874,219 : POKE 36875,219 : POKE
 36876,219 : POKE 36878,7
```

Cela produit le *la* du diapason à 440 Hz sur l'oscillateur 1, un *la* d'une octave plus haut sur l'oscillateur 2 et un *la* d'une autre octave plus haut sur l'oscillateur 3; tous ces sons sont réglés à un volume intermédiaire de 7. N'oubliez pas d'écrire un 0 dans chaque adresse pour les mettre hors fonction!

Notes et pauses

Si la durée des notes et le temps de pause entre chacune d'elles ne sont pas spécifiés, une séquence de notes ne devient qu'un son confus. Pour marquer ces périodes d'« attente » entre deux instructions POKE, deux méthodes peuvent être utilisées. La première consiste à insérer des boucles FOR ... NEXT qui forment des boucles vides de temporisation comme celle-ci :

```
10 POKE 36878,7
20 POKE 36876,203
30 FOR P = 1 TO 200
40 NEXT P
50 POKE 36878,0
60 POKE 36876,0
```

Cette séquence de commandes produit la note *ré* dièse pendant deux cents pas FOR ... NEXT. Cependant, cette méthode nécessite une synchronisation très précise de la boucle employée. Il est plus facile et plus rationnel de définir les durées et les pauses à l'aide de l'horloge interne du Vic-20 dont l'unité de mesure est 1/60 de seconde et qui peut être appelée à l'aide de la variable TI. Cela est extrêmement pratique pour construire une commande :

```
10 POKE 36878,7
20 POKE 36876,203 : D = TI
30 IF TI - D < 15 THEN 30
40 POKE 36878,0
50 POKE 36876,0
```

Ces instructions jouent la même note que précédemment mais pendant une période de 1/4 de seconde. D est défini à la valeur de TI lorsque le son est produit. A la ligne 30, le programme attend pendant 15/60 de seconde avant de poursuivre à la ligne 40. Des airs peuvent être construits en utilisant le même principe.

Affichage sur le Dragon

... et les possibilités graphiques du Dragon 32

L'ordinateur Dragon 32 offre une version particulière du BASIC nommée « BASIC Microsoft couleur étendu ». Plusieurs autres ordinateurs sur le marché utilisent cette version du BASIC, notamment la gamme des ordinateurs couleur Tandy. Le BASIC Microsoft est facile à utiliser et possède une gamme importante de commandes servant à dessiner des lignes, des cercles et autres formes géométriques. Une fois tracées, ces formes peuvent être colorées pour obtenir des affichages intéressants sans véritable effort de programmation.

Le Dragon 32 a sept niveaux de résolution, ce qui permet à l'utilisateur de travailler avec 512 points individuels au niveau le plus bas, et avec 49 152 points au niveau le plus haut. Huit couleurs sont disponibles; mais le choix peut être limité à quatre ou même à deux lorsqu'on travaille en haute résolution.

Modes de résolution

L'affichage normal de 16 lignes par 32 colonnes est le niveau de résolution le plus bas et la commande PRINT @ permet de placer un caractère dans l'une des 512 adresses d'écran.

L'autre mode de résolution divise l'écran en 32 lignes de 64 colonnes. La taille de chaque carré équivaut alors au quart de celle d'un caractère normal. Les points de cette dimension peuvent être tracés à l'écran au moyen de la commande SET et peuvent être effacés par la commande RESET.

Ces deux modes peuvent être affichés simultanément et se nomment modes « texte basse résolution ». Il existe également cinq niveaux d'affichage haute résolution, mais ceux-ci ne peuvent être affichés simultanément avec les affichages basse résolution. Les cinq modes haute résolution offrent la possibilité de choisir différents niveaux de résolution et différentes quantités de couleurs disponibles, et sont sélectionnés à l'aide de la commande PMODE.

PMODE	Résolution	Couleurs disponibles
0	128 * 96	2
1	128 * 96	4
2	128 * 192	2
3	128 * 192	4
4	256 * 192	2

Toute augmentation de la résolution doit évidemment se payer au niveau de la couleur et de la mémoire nécessaire pour stocker l'information écran et doit donc être prise en compte.

Bien qu'il existe peu de couleurs disponibles en haute résolution, le Dragon dispose d'une fonction permettant de choisir l'un des deux jeux de couleurs. La commande SCREEN permet cette sélection. Par exemple, SCREEN 1,0 sélectionne une haute résolution et le jeu de couleur 0. SCREEN 1,1 sélectionne de nouveau une haute résolution, mais, cette fois, l'autre jeu de couleurs est utilisé.

PAINT
Cette commande, très pratique, aide le programmeur à produire des images très intéressantes. La commande PAINT demande à l'ordinateur de colorer à partir d'un point donné une zone délimitée par une ligne de contour. Cela signifie que les cercles et les triangles (ou toute autre surface fermée) peuvent être colorés simplement.

DRAW
DRAW imite le mouvement d'un crayon sur l'écran, et permet à l'utilisateur de dessiner des lignes dans chacune des quatre directions. DRAW permet également d'agrandir ou d'orienter différemment l'image créée.

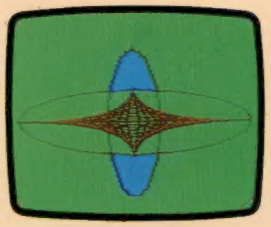
GET et **PUT**
GET demande à l'ordinateur de stocker un affichage en mémoire et PUT demande d'afficher un écran mis en mémoire.

PSET et **PRESET**
Ces commandes sont les équivalents haute résolution de SET et de RESET, présentées auparavant. Elles ont pour effet d'afficher ou d'effacer un point particulier sur l'écran. La couleur du point peut également être déterminée.

LINE
La commande LINE relie deux points spécifiés au moyen d'une ligne droite en haute résolution.

CIRCLE
CIRCLE permet de dessiner des cercles à haute résolution à partir d'un centre et d'un rayon donnés. Des fractions de cercles peuvent aussi être dessinées pour former des arcs et la forme circulaire peut être comprimée pour produire des ellipses.

Le Dragon est un ordinateur d'un prix raisonnable. Il offre de nombreuses commandes évoluées facilitant la programmation graphique. Il est plus destiné à des applications impliquant des affichages statiques qu'à celles qui nécessitent des affichages à action rapide. Les commandes du mode haute résolution, en particulier, font de cette machine un outil idéal pour l'enfant avide de découvertes. Le principal inconvénient du Dragon est l'impossibilité d'afficher simultanément du texte et des graphiques haute résolution.



Commande couleur
Cet affichage illustre bien les effets graphiques réalisables sur le Dragon grâce à quelques-unes de ses commandes évoluées. (Cl. Ian McKinnell.)

Haute résolution
Voici un court programme pour le Dragon 32 qui illustre certaines de ses possibilités graphiques haute résolution.
Le programme utilise PMODE 3; il ne s'agit pas du mode le plus élevé, mais il permet de faire une certaine utilisation de la couleur.

```
10 PCLS:PMODE3,1
20 SCREEN 1,0
30 COLOR 0, 1
40 FOR X=0 TO 127 STEP 10
50   LINE(X,85)-(127,85-X/3),
   PSET
60   LINE(X,85)-(127,85-X/3),
   PSET
70   LINE(255-X,85)-(127,85-
   X/3),PSET
80   LINE(255-X,85)-(127,85-
   X/3),PSET
90 NEXT X
100 CIRCLE(127,85),128,4,0.3
110 CIRCLE(127,85),30,4,3
120 PAINT(130,30),3,4
130 PAINT(130,130),3,4
140 GOTO 140
150 END
```

Le tri

La possibilité de trier l'information est essentielle dans la plupart des programmes, et il y a plusieurs façons de le faire.

Le tri est l'une des opérations les plus largement exécutées en informatique, mais c'est une des tâches où l'ordinateur est le moins efficace. Selon des données statistiques établies par des équipes de recherche, entre 3 % et 40 % du temps de traitement est réservé aux opérations de tri. Si l'on additionne les tâches associées à cette activité, comme la fusion des données et la recherche d'éléments spécifiques, ce chiffre dépasse certainement 50 %.

Les programmeurs ont probablement passé autant de temps à inventer des algorithmes de tri (méthodes générales de solution aux problèmes) que les ordinateurs à effectuer le tri lui-même. Les méthodes de tri évoluées sont extrêmement difficiles à analyser, mais il est assez facile de comprendre les méthodes les plus simples utilisées par les ordinateurs en partant d'un exemple de tri d'un jeu de cartes.

Étalez treize cartes d'une même couleur sur une table. Alignez-les, sans ordre particulier, mais en veillant à ce que l'as et le deux ne soient pas placés à l'extrême droite de la ligne. Les cartes doivent être triées en ordre décroissant (roi, dame, valet... as), en commençant à gauche. Comment résoudre ce problème?

a) Mettez comme marqueur une pièce de monnaie sous la carte située à l'extrême gauche afin de vous souvenir du point où vous en êtes. Comparez la carte marquée avec la carte située à sa droite. Sont-elles en ordre décroissant? Si elles ne le sont pas, permutez ces deux cartes en ne bougeant pas la pièce de monnaie, en ne bougeant qu'une carte à la fois et en ne plaçant pas les cartes l'une sur l'autre. Notez ce que vous devez faire pour effectuer cette permutation.

b) Quand les deux cartes sont en ordre, déplacez la pièce de monnaie d'une position vers la droite et répétez l'étape 1. Vous êtes maintenant dans une boucle qui se terminera lorsque vous placerez la pièce sur la position située à l'extrême droite. Quand vous atteignez cette position, vous venez de terminer une « passe ».

c) A la fin du premier examen, jetez un coup d'œil sur les cartes. L'as, qui est la carte la plus basse de la couleur, a finalement atteint sa position à l'extrême droite de la ligne. Si vous effectuez une nouvelle passe (comme aux étapes *a* et *b*), le deux sera placé à la bonne position. Ce processus sera répété jusqu'à ce que la couleur soit triée au complet en ordre décroissant.

Vous avez sans doute noté plusieurs inconvénients de cette méthode. Elle est assez laborieuse; elle n'est pas économique, puisque la seule permutation de deux cartes nécessite trois opérations différentes; et, en plus, plusieurs des comparaisons effectuées ne sont pas nécessaires. Par exemple, après la première passe, l'as est en bonne position, il n'est donc plus nécessaire de déplacer la pièce de monnaie jusqu'à la position 13 (où, de toute façon, aucune comparaison n'est nécessaire). Lors de la deuxième passe, la carte située à droite est en bonne position. Il n'est pas nécessaire de déplacer le pointeur à la position 12. En général, chaque passe se terminera à une position à droite de la position finale de la passe précédente.

Savoir où arrêter est un autre problème. Un ordinateur continuera à comparer les cartes indéfiniment jusqu'à ce qu'on lui dise d'arrêter. La seule règle sûre est la suivante : arrêter après une passe sans permutation. En d'autres mots, si vous avez parcouru toutes les données sans en modifier l'ordre, celles-ci doivent être dans l'ordre désiré.

Cette méthode effectue un tri progressif. Ses principaux avantages sont les suivants : elle implique des techniques simples de programmation, utilise très peu de mémoire supplémentaire et est raisonnablement efficace pour de petites quantités de données partiellement ordonnées. Ce sont des critères selon lesquels un algorithme de tri doit être jugé, bien que, lorsque la quantité de données est importante, la vitesse puisse devoir être sacrifiée à l'économie de mémoire :

a) Brassez les cartes et étalez-les de nouveau en plaçant une pièce de 20 centimes sous la deuxième carte à partir de la gauche. Quelle que soit la carte posée sur la pièce de 20 centimes au début de chaque passe, nous l'appellerons la « carte 20 centimes ».

b) Poussez la carte 20 centimes en dehors de la ligne, laissez un espace, et placez une pièce de 1 franc sous la carte située immédiatement à gauche. Nommez cette carte la « carte 1 franc ».

c) Comparez la carte 20 centimes avec la carte 1 franc. Si elles sont en ordre, ramenez la carte 20 centimes en place et passez à l'étape *d*. Si elles ne sont pas en ordre, poussez la carte 1 franc dans l'espace et déplacez la pièce de 1 franc d'une position vers la gauche pour marquer une nouvelle carte 1 franc (si la carte 1 franc est située à l'extrême gauche, cela ne s'appliquera pas; placez donc la carte 20 centimes sur l'espace et passez à l'étape *d*).

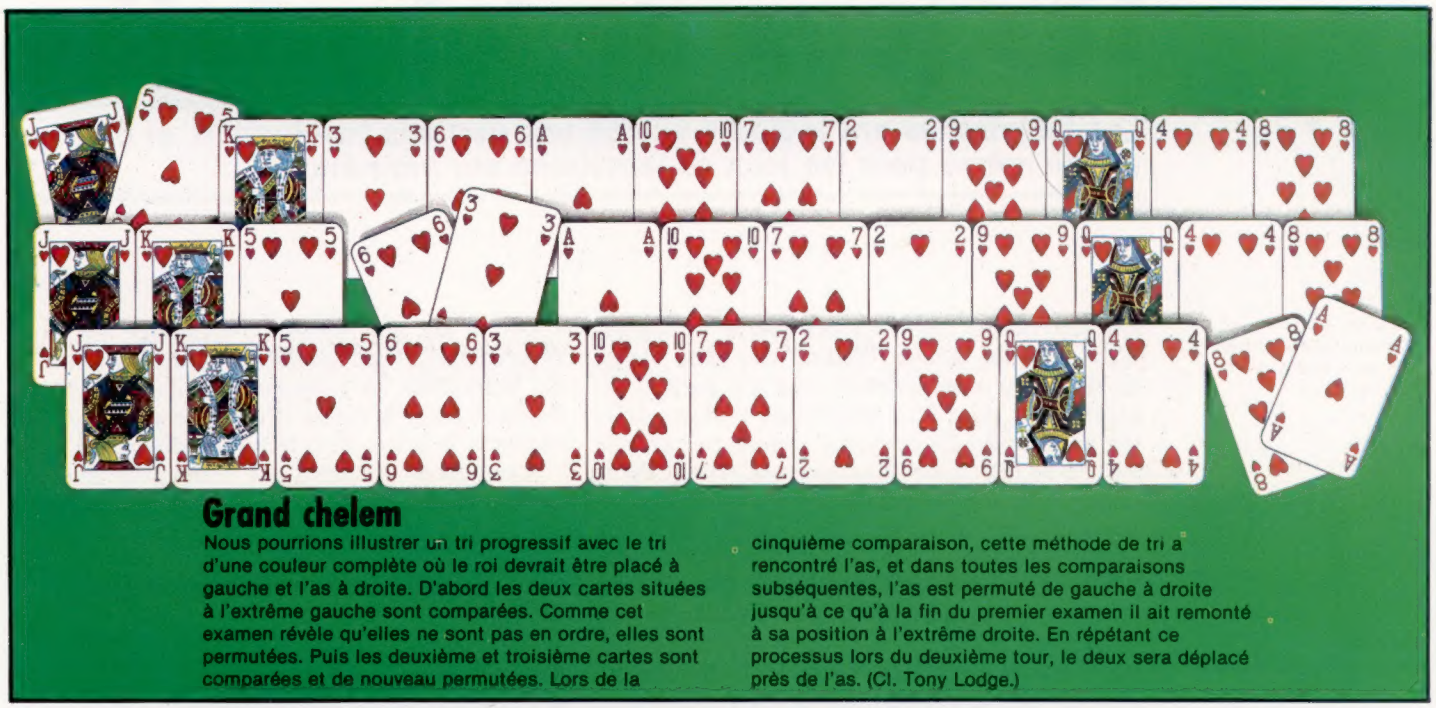
Comparez cette carte 1 franc avec la carte 20 centimes (la carte déplacée). Maintenant répétez l'étape *c* jusqu'à ce que la bonne position de la carte 20 centimes soit trouvée.

Tri progressif
Ce diagramme illustre le tri progressif pour une main de neuf cartes (D est la carte du dix). Le tri s'effectue à partir du côté droit à chaque examen. Le 1 et le 2 sous la main de cartes indiquent quelles cartes sont actuellement comparées.

```
Début tri
2 8 3 9 D 5 R 6 7
début examen 1
1 2
8 2 9 3 D 5 R 6 7
  1 2
8 9 2 3 D 5 R 6 7
    1 2
8 9 3 2 D 5 R 6 7
      1 2
8 9 3 D 2 5 R 6 7
        1 2
8 9 3 D 5 2 R 6 7
          1 2
8 9 3 D 5 R 2 6 7
            1 2
8 9 3 D 5 R 6 2 7
              1 2
8 9 3 D 5 R 6 7 2  Fin revue 1
9 8 D 5 R 6 7 3 2  Fin revue 2
9 D 8 R 6 7 5 3 2  Fin revue 3
D 9 R 8 7 6 5 3 2  Fin revue 4
D R 9 8 7 6 5 3 2  Fin revue 5
R D 9 8 7 6 5 3 2  Fin revue 6
Fin tri
```

Tri par insertion
Avec le tri par insertion, le tri s'effectue à partir du côté gauche. Les cartes sont déplacées directement à leur position correcte lors de leur examen.

```
Début tri
2 1
8 2 9 3 D 5 R 6 7
2  1
9 8 2 3 D 5 R 6 7
    2 1
9 8 3 2 D 5 R 6 7
2      1
D 9 8 3 2 5 R 6 7
      2  1
D 9 8 5 3 2 R 6 7
2          1
R D 9 8 5 3 2 6 7
          2    1
R D 9 8 6 5 3 2 7
          2      1
R D 9 8 7 6 5 3 2
Fin tri
```



Grand chelem

Nous pourrions illustrer un tri progressif avec le tri d'une couleur complète où le roi devrait être placé à gauche et l'as à droite. D'abord les deux cartes situées à l'extrême gauche sont comparées. Comme cet examen révèle qu'elles ne sont pas en ordre, elles sont permutées. Puis les deuxième et troisième cartes sont comparées et de nouveau permutées. Lors de la cinquième comparaison, cette méthode de tri a rencontré l'as, et dans toutes les comparaisons subséquentes, l'as est permuté de gauche à droite jusqu'à ce qu'à la fin du premier examen il ait remonté à sa position à l'extrême droite. En répétant ce processus lors du deuxième tour, le deux sera déplacé près de l'as. (Cl. Tony Lodge.)

d) Déplacez la carte 20 centimes d'une position vers la droite et répétez les étapes *b* et *c*. Lorsque vous ne pouvez déplacer la pièce de 20 centimes plus loin vers la droite, les cartes sont triées.

Cela se nomme un « tri par insertion », et ressemble beaucoup à la méthode que les gens utilisent pour trier des cartes. Bien que cette méthode soit plus difficile à programmer que le tri progressif, elle est beaucoup plus efficace. Ultérieurement, nous examinerons des algorithmes de tri de données plus complexes.

```
  9 REM ********************************
 10 REM * ALGORITHMES DE TRI *       *
 11 REM ********************************
100 IMPUT " COMBIEN D'ITEMS A TRIER ? " ; LT
150 IF LT < 3 THEN LET LT = 3
200 LT LT = INT (LT)
250 DIM R(LT), C(LT)
300 LET Z = 0 : LET Q = 0 : LET P = 0
350 LET I = 1 : LET 0 = 0 : LET II = 2 : LET TH = 2
400 INPUT " COMBIEN DE TESTS ? " ; N
450 FOR CT = I TO N
500 GOSUB 4000
550 FOR SR = I TO TH
600 GOSUB 5000
650 PRINT : PRINT : PRINT : PRINT
700 PRINT " TEST N° " ; CT + SR/10
750 INPUT " APPUYEZ SUR RETURN POUR COMMENCER TRI " ; A$
800 PRINT " LA LISTE NON TRIEE EST "
850 GOSUB 3000
900 ON SR GOSUB 6000, 7000
950 PRINT " LA LISTE TRIEE EST "
1000 GOSUB 3000
1050 NEXT SR
1100 NEXT CT
1150 END
2999 REM ********************************
3000 REM *   IMPRESSION DE LA LISTE   *
3001 REM ********************************
3100 FOR K = I TO LT
3200 PRINT R(K) ;
3300 NEXT K
3400 PRINT
3500 RETURN
3999 REM ********************************
4000 REM *    GENERATEUR ALEATOIRE    *
4001 REM ********************************
4100 RANDOMIZE
4200 FOR K = I TO LT
4300 LET C(K) = INT(100 * RND)
4400 NEXT K
4500 RETURN
4999 REM ********************************
5000 REM *    GENERATEUR ALEATOIRE    *
5001 REM ********************************
5100 FOR K = I TO LT
5200 LET R(K) = C(K)
5300 NEXT K
5400 PRINT : PRINT
5500 RETURN
5999 REM ********************************
6000 REM *          PROGRESSIF          *
6001 REM ********************************
6050 PRINT " TRI PROGRESSIF - DEPART ! "
6100 FOR P = LT - I TO I STEP - I
6150 LET F = - I
6200 FOR Q = I TO P
6250 LET Z = Q + I
6300 IF R(Q) < R(Z) THEN LET D = R(Q) :
     LET R(Q) = R(Z) : LET R(Z) = D : LET F = 0
6350 NEXT Q
6400 IF F = - I THEN LET P = I
6450 NEXT P
6500 PRINT " TRI PROGRESSIF - STOP ! "
6550 RETURN
6999 REM ********************************
7000 REM *           INSERTION          *
7001 REM ********************************
7050 PRINT " TRI INSERTION - DEPART ! "
7100 FOR P = II TO LT
7200 LET D = R(P)
7300 FOR Q = P TO II STEP - I
7400 LET R(Q) = R(Q - I)
7500 IF D <= R(Q) THEN LET R(Q) = D : LET Q = II
7600 NEXT Q
7700 IF D > R (I) THEN R (I) = D
7800 NEXT P
7850 PRINT " TRI PAR INSERTION - STOP ! "
7900 RETURN
```

Tri à haute vitesse

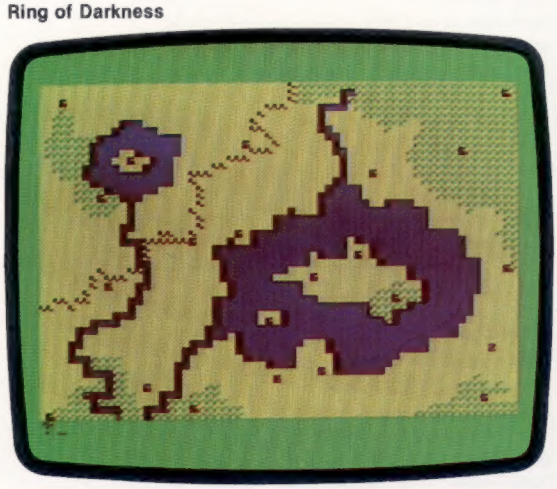
Ce programme BASIC démontre la différence existant entre un tri progressif et un tri par insertion. Le programme a été écrit en visant l'efficacité et la vitesse. Nous n'avons donc pas documenté le fonctionnement des routines. Le listing devrait fonctionner sur la plupart des machines, mais consultez les variantes de l'instruction ON... GOSUB et celles de RND et de RANDOMIZE.

Les jeux de labyrinthe

Les labyrinthes ont toujours exercé une certaine fascination, et il en est de même pour les jeux de labyrinthe sur ordinateur.

Les labyrinthes ont toujours éveillé l'imagination des plus jeunes comme des plus vieux, qu'ils soient assez grands pour s'y perdre, ou assez petits pour être tenus dans le creux de la main. Mais depuis quelques années, une grande variété de jeux sur ordinateur sont fondés sur le principe du labyrinthe, depuis une simple représentation bidimensionnelle jusqu'à des jeux ayant des implications tridimensionnelles complexes. Dans ce dernier type de jeu, le programme simule un déplacement à l'intérieur du labyrinthe, et le joueur a vraiment l'impression d'y marcher. Pour l'aider, ou pour créer encore plus de confusion, certains de ces programmes tridimensionnels affichent brièvement une vue aérienne du labyrinthe.

Ring of Darkness

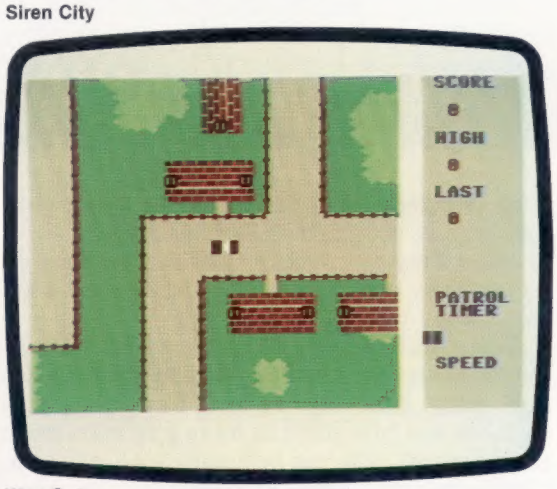


Comme les effets sonores et visuels deviennent plus sophistiqués, les programmeurs ont pu laisser libre cours à leur imagination. Un joueur désirant uniquement faire une petite promenade de santé dans un labyrinthe devrait éviter ceux qui cachent diverses créatures monstrueuses. 3D Glooper fait partie de cette catégorie de jeux; le joueur recherche des carreaux spéciaux et peut être attaqué à tout moment par des monstres. L'arrivée imminente de ces créatures est cependant annoncée par le bruit caractéristique de leurs mâchoires. Atic Atac (Spectrum) est un jeu de labyrinthe très animé dans lequel le joueur joue le rôle de trois personnages différents. Le labyrinthe est formé d'une série de tunnels à niveaux multiples, d'escaliers et de grandes salles, où vous effectuez une course contre la montre. Les grandes salles sont habitées par des créatures et des objets étranges.

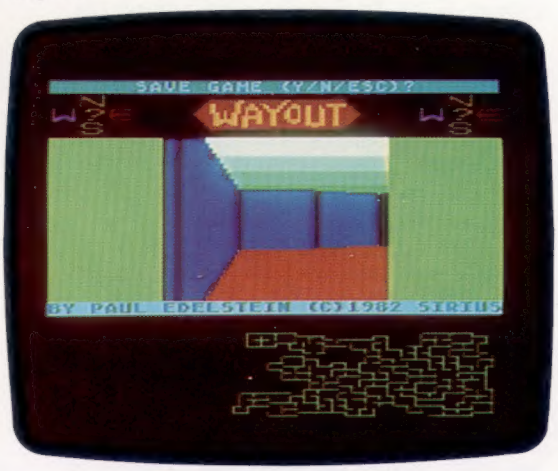
Way Out est un programme qui simule avec une incroyable vérité les tentatives destinées à sortir d'un labyrinthe. L'affichage est une représentation véritablement tridimensionnelle du labyrinthe, et vous commandez vos divers déplacements à l'aide d'une manette qui fait déplacer la scène dans la bonne direction.

Examinons certaines des techniques de programmation fondamentales impliquées dans la conception de jeux de ce type.

Siren City



Way Out



Siren City
Ce jeu sur Commodore 64 est un jeu traditionnel avec vue aérienne perfectionnée. Une voiture de police patrouille la ville.

Ring of Darkness
Bien que ce jeu exécuté sur le Dragon soit un véritable jeu d'aventures, il possède comme principal élément un labyrinthe tridimensionnel. Des fosses et des échelles vous permettent de monter et de descendre.

Way Out
Une image tridimensionnelle d'une vérité impressionnante est obtenue sur un Spectrum avec Way Out. Un mouvement à peine perceptible de la manette de jeu modifie la perspective de l'affichage. (Cl. Ian McKinnell.)

Construction de labyrinthes

La façon habituelle de stocker l'information relative à un labyrinthe est d'utiliser un tableau à deux dimensions — L$ (LIGNE,COLONNE) par exemple. Chaque cellule du tableau définit les caractéristiques d'une cellule du labyrinthe. Vous pourriez par exemple utiliser une chaîne de quatre caractères pour représenter le sud, l'ouest, le nord et l'est. Zéro indiquerait l'absence d'un mur et un sa présence. Donc, si L$(5,6) renferme

la chaîne « 1011 », cela signifierait que la cellule, sur la ligne cinq, colonne six, est fermée par des murs au sud, au nord et à l'est.

Pour économiser l'espace mémoire, le tableau peut être numérique, et le nombre à quatre chiffres peut alors être considéré comme un nombre binaire. Dans notre exemple précédent, la cellule possédant des murs nord, est et sud renfermerait le nombre 11 (1011).

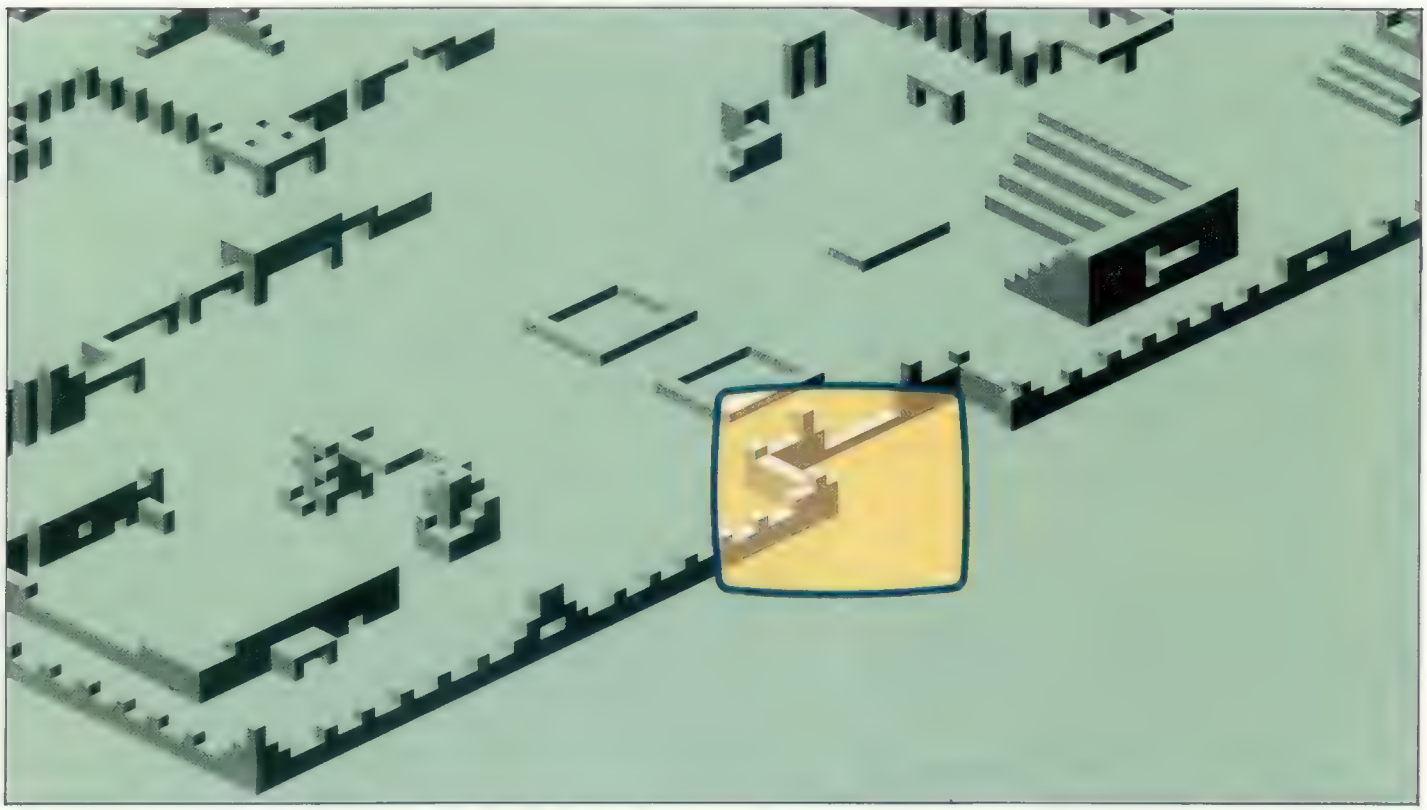
Toutes les cellules auraient quatre murs au départ. En créant une entrée de façon aléatoire en un point du périmètre, la cellule suivante serait choisie au hasard parmi l'une des cellules adjacentes. Lorsque cette cellule est choisie, la séquence continue par une sélection aléatoire de l'une des trois cellules adjacentes, en ne considérant pas la cellule d'origine.

Si toutes les cellules adjacentes ont déjà été visitées, le programme doit revenir à la cellule précédente pour effectuer un nouveau branchement.

Une autre méthode de définition des caractéristiques d'une cellule serait d'utiliser une numérotation binaire plus sophistiquée, particulièrement indiquée pour afficher des perspectives tridimensionnelles. Il y a seize façons différentes de construire une cellule : sans mur, avec un seul mur (quatre possibilités), avec des murs opposés (deux), avec deux murs adjacents (quatre), avec trois murs adjacents (quatre).

En utilisant le nombre binaire approprié (0 à 15) pour chacune de ces situations, il est possible de « faire tourner » le nombre vers la gauche ou vers la droite pour obtenir la représentation appropriée à présenter au joueur. Par exemple, un mur nord pourrait être représenté par 2 (0010), un mur sud par 8 (1000), un mur est par 1 (0001) et un mur ouest par 4 (0100). Si le joueur faisant face au nord dans une pièce ne possédant qu'un mur à l'ouest (4) se tourne pour faire face à l'ouest, il verrait alors une pièce avec un mur au nord (car ce qui fait face au joueur dans un affichage tridimensionnel est toujours dit être « au nord »). Comme le joueur s'est tourné vers sa gauche (l'ouest), déplacer la configuration binaire d'une position vers la droite nous donne la description désirée, c'est-à-dire que le nombre binaire 0100 (décimale 4) du mur ouest devient le nombre binaire 0010 (décimale 2 — un mur nord !). Les bits sont déplacés dans la direction inverse quand le joueur se tourne vers la droite. Ils sont déplacés de deux positions pour faire un demi-tour. Il est évidemment nécessaire de prévoir un système de « bouclage » pour ne pas perdre, pendant ce processus, les bits des extrémités gauche et droite du demi-octet, sinon les caractéristiques d'identification de la cellule seraient modifiées à chaque fois que le joueur effectue une rotation à l'intérieur. Une cellule, définie initialement 0011 (murs nord et est), doit devenir 0110 si le joueur se tourne vers la droite, et 1100 s'il fait demi-tour.

En code machine, il existe des instructions spéciales pour « tourner » les chiffres binaires vers la gauche ou vers la droite. En BASIC, un nombre binaire à 4 bits, compris entre 0 et 15, peut être décalé vers la gauche en multipliant le nombre par 2, et en soustrayant 15 si le résultat est supérieur à 15. Pour tourner vers la droite : diviser par 2 s'il s'agit d'un nombre pair, ou additionner 15 et diviser par 2 s'il s'agit d'un nombre impair.



Ant Attack
Lorsque ce jeu est exécuté sur le Spectrum, l'écran de l'ordinateur devient une fenêtre sur un immense terrain de jeu de type labyrinthe. Vous découvrez progressivement la scène du jeu au cours de son déroulement.

L'Aquarius

Fabriqué par un constructeur de jouets très connu, l'Aquarius n'en est pas moins un ordinateur très sérieux et... bon marché.

Avec son processeur Z80 et son type de clavier, l'Aquarius se range dans la même catégorie de micro-ordinateurs que le Spectrum. C'est cependant une machine beaucoup plus souple, surtout en raison de son bus d'extension intégré qui a été bien exploité par ses concepteurs.

Grâce à ce bus, de nombreux modules d'extension peuvent être connectés, allant des modules RAM de 4 K jusqu'à un grand châssis d'extension. Parmi ces dispositifs, le plus pratique est le « petit châssis d'extension » qui possède deux connecteurs pour des modules de mémoire additionnelle ou de programme, ainsi que deux canaux sonores et deux commandes manuelles à disque.

Le clavier et l'affichage de l'Aquarius n'ont pas la qualité des plus grosses machines. Il n'y a pas de barre d'espacement, et la réponse des touches n'est pas très sensible. Ce clavier ne permet donc pas une frappe très rapide. L'écran de 24 lignes par 40 caractères, bien plus grand que sur d'autres ordinateurs, n'est pas approprié à une petite application de gestion.

L'affichage offre seize couleurs qui peuvent être utilisées pour le texte ou pour l'arrière-plan. Bien qu'il n'offre pas la possibilité de définir ses propres caractères, il possède 256 symboles affichables, dont des lettres minuscules et majuscules, et toute une variété de symboles graphiques. Il peut créer des affichages d'une résolution de 320 × 192 points. L'affichage s'effectue au moyen d'un téléviseur. La sortie sur un moniteur n'a pas été prévue. La qualité de l'affichage est moyenne, avec une tendance évidente vers les tons bleus et des caractères légèrement flous; par contre, l'affichage est stable et d'une brillance suffisante et offre une bonne gamme de couleurs.

Mini extension
Ce périphérique possède deux ports de cartouches permettant de connecter simultanément une cartouche programme et un module de mémoire. Il offre également deux commandes manuelles et trois canaux sonores additionnels. (Cl. Chris Stevens.)

Imprimante Aquarius
Peu coûteuse, elle utilise un mécanisme thermique et nécessite donc un papier thermique spécial. Elle peut imprimer à une vitesse de 80 caractères par seconde, sur une largeur totale de 40 colonnes.

Cette machine peut produire des sons, bien qu'elle n'offre pas les commandes sophistiquées d'enveloppe et de forme d'onde que l'on retrouve sur d'autres ordinateurs.

Un des dispositifs complémentaires les plus intéressants prévus pour l'Aquarius est le système BSR X-10 qui pourra commander de nombreux appareils ménagers. Ce système permet de commander jusqu'à deux cent cinquante-cinq appareils électriques différents au moyen de signaux générés par l'unité centrale! Aucun montage additionnel n'est nécessaire, puisque ces signaux sont fournis sous la forme d'impulsions dans le système électrique. Ces impulsions ne sont pas assez importantes pour modifier de façon sensible le courant du secteur, mais un détecteur X-10 branché dans toute prise murale du secteur peut recevoir le code et modifier le courant fourni à l'appareil domestique qu'il commande conformément à la demande.

L'unité contrôleur est programmée par l'Aquarius selon des cycles hebdomadaires. Pendant cette opération, l'ordinateur n'est pas disponible pour d'autres utilisations. Si le programme de préréglage est satisfaisant, l'ordinateur est libre à tout autre moment pour une utilisation ordinaire.

Le clavier de l'Aquarius
Le clavier de l'Aquarius est l'un de ses points faibles. Annoncé comme ayant une disposition QWERTY standard, il ne mérite pas vraiment ce titre. Il n'y a pas de barre d'espacement, il n'a qu'une seule touche SHIFT, et la touche RETURN est placée à une position non conventionnelle; de plus, l'espacement entre les touches n'est pas du tout celui d'une machine à écrire.

Connecteur RF
La sortie compatible téléviseur est fournie ici. L'utilisation d'un moniteur n'est pas prévue.

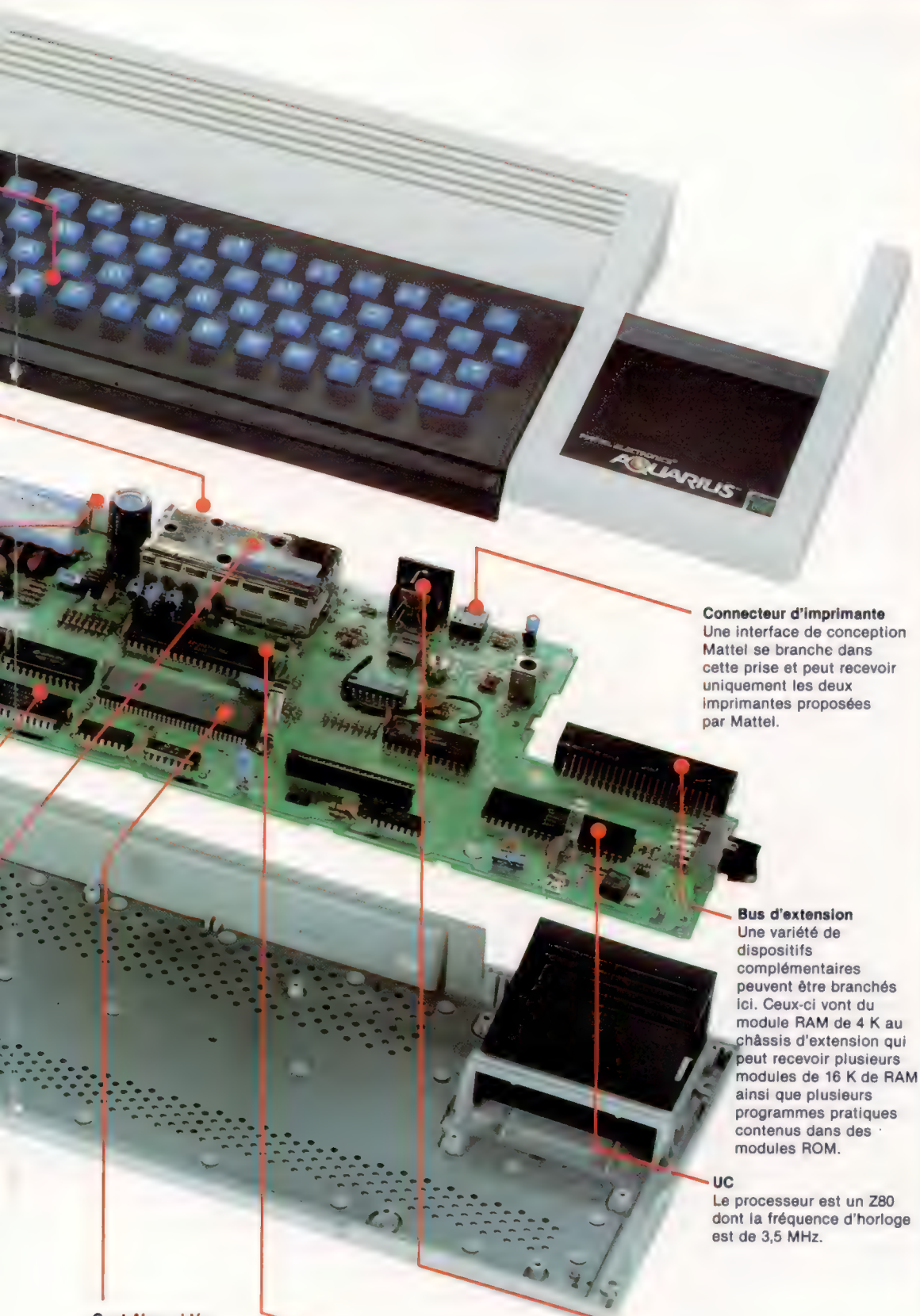
Connecteur d'alimentation
L'alimentation est fournie à partir d'un petit transformateur.

RAM
La mémoire utilisateur intégrée de 4 K est contenue dans ces quatre puces.

Chris Stevens

ROM
Le BASIC Microsoft standard 8 K est contenu dans ces puces. Les extensions qui ont été apportées pour gérer les graphiques et le son occupent le reste de l'espace en ROM.

Modulateur
Le signal d'affichage est converti en un signal téléviseur standard.



Connecteur d'imprimante
Une interface de conception Mattel se branche dans cette prise et peut recevoir uniquement les deux imprimantes proposées par Mattel.

Bus d'extension
Une variété de dispositifs complémentaires peuvent être branchés ici. Ceux-ci vont du module RAM de 4 K au châssis d'extension qui peut recevoir plusieurs modules de 16 K de RAM ainsi que plusieurs programmes pratiques contenus dans des modules ROM.

UC
Le processeur est un Z80 dont la fréquence d'horloge est de 3,5 MHz.

Contrôleur vidéo
La conception des composants électroniques qui commandent l'affichage vidéo est maintenant l'aspect le plus important de la conception d'un ordinateur. Cette puce de contrôle est plus grosse que le microprocesseur lui-même.

Puce de sécurité
Cette puce construite sur mesure a pour but de rendre très difficile pour d'autres fabricants la réalisation de cartouches de programme exécutables sur l'Aquarius.

Connecteur de cassette
L'interface de cassette est une prise DIN et possède des connexions pour commander le moteur de l'unité à cassette.

AQUARIUS

PRIX

moins de 1 000 F

DIMENSIONS

345 × 150 × 55 mm.

VITESSE DE L'HORLOGE

3,5 MHz.

MÉMOIRE

10 K de ROM ; plus 4 K de RAM, avec extension possible jusqu'à 64 K.

AFFICHAGE VIDÉO

24 lignes de 40 caractères, 16 couleurs avec arrière-plan et premier plan définis indépendamment ; 256 caractères prédéfinis, mais l'utilisateur ne peut définir ses propres caractères.

INTERFACES

Cassette, imprimante, bus d'extension.

LANGAGE INTÉGRÉ

BASIC Microsoft.

AUTRES LANGAGES DISPONIBLES

Un BASIC Microsoft étendu et un Aquarius LOGO ont été promis par Mattel. Ceux-ci seront livrés sous la forme de modules ROM.

ACCESSOIRES FOURNIS

Manuel d'installation et manuel BASIC, fil de téléviseur.

CLAVIER

49 boutons-poussoirs. Le bouton de remise à zéro est encastré afin qu'il ne puisse être pressé accidentellement.

DOCUMENTATION

La documentation est particulièrement bonne pour des débutants, avec un ensemble pratique de cartes de référence qui décrivent les fonctions principales de la machine et du BASIC intégré. Il y a pénurie de renseignements techniques, mais si l'on tient compte du marché visé par l'Aquarius, la documentation de cet ordinateur peut servir d'exemple.

Branchements

Lors du développement d'un programme, sa structure prend de plus en plus l'apparence d'un arbre, et de nouvelles branches apparaissent à chaque étape de son perfectionnement.

Dans le dernier article, nous avons examiné certains des problèmes concernant la recherche d'un élément spécifique au sein d'une liste — en supposant que la liste avait déjà été triée selon un certain ordre. Nous reviendrons sur ce sujet lorsque le moment sera venu de commencer à écrire des routines de recherche. Entre-temps, nous continuerons à développer le thème de la programmation descendante tout en produisant un code pour les deux parties du programme principal :

```
PROGRAMME PRINCIPAL
DÉBUT
  INITIALISATION (procédure)
  ACCUEIL (procédure)
  CHOIX (procédure)
  EXÉCUTION (procédure)
FIN
```

La première procédure, * INITIALISATION *, impliquera des interventions assez complexes — mise en place de tableaux, lecture de données dans ces tableaux, exécution de diverses vérifications, et ainsi de suite —, et nous négligerons pour l'instant les détails de cette procédure. Les deux autres parties du programme principal sont les procédures ACCUEIL et CHOIX. Lors du développement de ces procédures, nous suggérons une méthodologie dont le but est d'éviter que les diverses interventions impliquées dans le processus de la programmation descendante n'entraînent une certaine confusion.

Comme le nombre de « branches » ne cesse d'augmenter (les procédures deviennent de plus en plus perfectionnées), elles prennent de plus en plus de place sur la feuille. Éventuellement, il devient impossible de tout écrire sur une page.

Il existe une méthode très efficace pour disposer la documentation d'un programme, c'est de numéroter systématiquement les étapes de son développement. Nous avons utilisé des chiffres romains pour indiquer le niveau de perfectionnement et des chiffres arabes pour indiquer la sous-section du programme. Une feuille distincte est utilisée pour chaque niveau de perfectionnement et les pages de chaque bloc ou module de programme peuvent être facilement conservées ensemble. Voici le système de numérotation de notre programme :

```
I PROGRAMME PRINCIPAL
DÉBUT
  1. INITIALISATION (procédure)
  2. ACCUEIL (procédure)
  3. CHOIX (procédure)
  4. EXÉCUTION (procédure)
FIN
```

Comme nous l'avons mentionné précédemment, nous ne nous préoccupons pas du développement d'INITIALISATION pour l'instant, et nous nous concentrons sur les procédures ACCUEIL et CHOIX.

```
II 2 (ACCUEIL)
DÉBUT
  1. Affichage d'un message d'accueil
  2. BOUCLE (jusqu'à ce que la barre d'espacement soit pressée)
     FINBOUCLE
  3. Appel de * CHOIX *
FIN
```

```
III 2 (ACCUEIL) 1 (affichage du message)
DÉBUT
  1. Effacer l'écran
  2. IMPRESSION du message d'accueil
FIN
```

```
III 2 (ACCUEIL) 2 (BOUCLE attendre barre d'espacement)
DÉBUT
  1. BOUCLE (jusqu'à ce que la barre d'espacement soit pressée)
       SI barre espacement pressée
       ALORS
     FINBOUCLE
FIN
```

```
III 2 (ACCUEIL) 3 (appel * CHOIX *)
DÉBUT
  1. ALLER SOUS-PROGRAMME * CHOIX *
FIN
```

Il doit être clair ici que **III-2-1** et **III-2-3** sont prêts à être traduits directement en BASIC, mais que **III-2-2** doit subir une autre étape de perfectionnement :

```
IV 2 (ACCUEIL) 2 (BOUCLE)
DÉBUT
  1. BOUCLE (jusqu'à ce que la barre d'espacement soit pressée)
       SI TOUCHE$ n'est pas espace ALORS continuer
     FIN BOUCLE
FIN
```

Nous avons maintenant atteint le moment où la formulation BASIC de la procédure ACCUEIL peut subir un autre perfectionnement :

```
IV 2 (ACCUEIL) 1 (affichage du message) CODE BASIC
REM SOUS-PROGRAMME * ACCUEIL *
PRINT
PRINT
PRINT
PRINT
PRINT TAB (16); « * VOICI * »
```

```
PRINT TAB(4); « * LE CARNET D'ADRESSES INFORMATISÉ * »
PRINT TAB(10); « * ABC INFORMATIQUE * »
PRINT
PRINT TAB(4) : « (APPUYEZ SUR LA BARRE D'ESPACEMENT) »
```

V2 (ACCUEIL) 2 (BOUCLE attendre barre espacement) CODE BASIC

```
LET L = 0
FOR L = 1 TO 1
IF INKEY$ <> « » THEN LET L = 0
NEXT L
```

IV 2 (ACCUEIL) 3 (appel de * CHOIX *) CODE BASIC

```
GOSUB * CHOIX *
RETURN
```

Notez que nous avons maintenant commencé à initialiser des variables dans les diverses routines que nous écrivons à l'aide d'instructions comme LET I = 0. Pas nécessaire dans certaines circonstances, il est préférable de prendre cette habitude si vous pouvez vous la rappeler, et si vous disposez d'assez d'espace en RAM. Il y a trois raisons de le faire :

a) La présence d'une liste d'instructions LET au début d'une routine vous aide à vous souvenir des variables locales qui sont utilisées.

b) Vous ne pouvez connaître de façon certaine quel a été le contenu laissé dans cette variable après la dernière exécution de la routine (cela n'a pas toujours d'importance).

c) L'insertion d'instructions comme LET I = 0 dans le bon ordre peut accélérer l'exécution d'un programme.

Nous avons modifié notre façon d'utiliser la boucle FOR ... NEXT pour simuler une structure EXÉCUTER ... JUSQU'À ou RÉPÉTER ... JUSQU'À. Au lieu d'utiliser FOR I = 0 TO 1 ou FOR I = 0 TP 1 STEP 0, nous utilisons maintenant FOR I = 1 TO 1. Cette instruction fonctionne sur tous les ordinateurs domestiques dont nous parlons régulièrement. Les autres méthodes impliquaient des modifications pour diverses machines. FOR I = 1 TO 1 ... NEXT I n'exécutera la boucle qu'une seule fois. Cependant, si I est mis à zéro quelque part à l'intérieur de la boucle, celle-ci sera exécutée de nouveau, et ainsi de suite. Nous pouvons insérer une instruction LET I = 0 si la condition de sortie n'est pas remplie, ou nous pouvons mettre I à zéro immédiatement après l'instruction FOR, et le forcer à 1 si la condition de sortie est satisfaite. Par conséquent, les deux boucles suivantes ont le même résultat :

```
FOR I = 1 TO 1
IF INKEY$ <> « » THEN LET I = 0
NEXT I
```

```
FOR I = 1 TO 1
LET I = 0
IF INKEY$ = « » THEN LET I = 1
NEXT I
```

Ce code BASIC représente tout ce qui est nécessaire pour le sous-programme ACCUEIL. Nous n'avons pas mentionné les numéros de ligne parce que nous ne pouvons pas vraiment le faire tant que tous les modules du programme ne sont pas prêts pour le codage final. Par exemple, nous ne connaissons pas encore à ce stade quels sont les numéros de ligne appropriés pour les commandes GOSUB. Si vous désirez tester le module à ce stade, il sera nécessaire de créer certains sous-programmes factices d'entrée et de sortie. Nous devons signaler l'utilisation de fonctions TAB et d'instructions « effacer l'écran » dans cette portion du programme. TAB déplace le curseur le long d'une ligne du nombre (l'argument) placé entre parenthèses. Les nombres que nous avons utilisés afficheront un message centré sur un écran de 40 caractères. Si votre affichage est moins large (par exemple, le Spectrum affiche 32 caractères par ligne) ou plus large (les ordinateurs plus gros affichent généralement 80 caractères par ligne), les arguments de ces fonctions TAB devront être modifiés. L'instruction servant à effacer l'écran est CLS dans de nombreuses versions de BASIC, mais la version Microsoft utilisée pour développer ce programme ne reconnaît pas cette instruction. Nous avons donc utilisé PRINT CHR$(12), puisque notre machine utilise ASCII 12 comme caractère d'« effacement écran » — d'autres machines utilisent ASCII 24 pour la même fonction.

```
10 REM PROGRAMME PRINCIPAL FACTICE
20 PRINT CHR$(12)
30 GOSUB 100
40 END
100 REM SOUS-PROGRAMME * ACCUEIL *
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT TAB(16); « * VOICI * »
160 PRINT TAB(4); « * LE CARNET D'ADRESSES INFORMATISÉ * »
170 PRINT TAB(10); « *ABC INFORMATIQUE * »
180 PRINT
190 PRINT TAB(4) : « (APPUYEZ SUR LA BARRE D'ESPACEMENT) »
195 LET L = 0
200 FOR L = 1 TO 1
210 IF INKEY$ <> « » THEN LET L = 0
220 NEXT L
230 PRINT CHR$(12)
240 GOSUB 1000
250 RETURN
1000 REM SOUS-PROGRAMME FACTICE
1010 PRINT « SOUS PROGRAMME FACTICE »
1020 RETURN
```

Nous utiliserons maintenant exactement la même approche pour perfectionner la procédure CHOIX.

II 3 (CHOIX)

```
DÉBUT
  1. AFFICHAGE MENU
  2. ENTRÉE DU CHOIX
  3. Appeler le sous-programme CHOIX
FIN
```

III 3 (CHOIX) 1 (AFFICHAGE du menu)

```
DÉBUT
  1. Effacer l'écran
  2. AFFICHAGE du menu
FIN
```

III 3 (CHOIX) 2 (ENTRÉE DU CHOIX)

```
DÉBUT
```

```
  1. ENTREE DU CHOIX
  2. Vérifier validité du choix
FIN
```

III 3 (CHOIX) 3 (appel de CHOIX)

```
DEBUT
  1. CHOIX MULTIPLE
     FINCHOIX
FIN
```

III-3-1 (AFFICHAGE du menu) peut maintenant être codé en BASIC :

IV 3 (CHOIX) 1 (AFFICHAGE du menu) CODE BASIC

```
REM EFFACEMENT ECRAN
PRINT CHR$(12) : REM OU « CLS »
PRINT
PRINT
PRINT
PRINT
PRINT « 1. TROUVER ENREGISTREMENT (A PARTIR DU NOM) »
PRINT « 2. TROUVER ENREGISTREMENT (AVEC NOM PARTIEL) »
PRINT « 3. TROUVER ENREGISTREMENT (A PARTIR DE LA VILLE) »
PRINT « 4. TROUVER ENREGISTREMENT (A PARTIR DES INITIALES) »
PRINT « 5. LISTER TOUS LES ENREGISTREMENTS »
PRINT « 6. AJOUTER UN NOUVEL ENREGISTREMENT »
PRINT « 7. MODIFIER UN ENREGISTREMENT »
PRINT « 8. EFFACER UN ENREGISTREMENT »
PRINT « 9. QUITTER ET SAUVEGARDER »
```

III-3-2 (ENTRÉE CHOIX) et **III-3-3 (appel de CHOIX)** doivent être mieux définis. Examinons d'abord le prochain niveau de perfectionnement de **III-3-2**.

Affecter une valeur numérique à la variable CHOIX est très simple : après le message de sollicitation, la commande INPUT CHOIX se charge de cette opération. Cependant, il n'y a que neuf choix possibles. Que se passerait-il si nous entrions par erreur 0 ou 99? Puisque le CHOIX que nous faisons détermine quelle partie du programme sera appelée par la suite, il est nécessaire de se prémunir contre une erreur d'entrée. Nous devons donc insérer une procédure de vérification de l'entrée. C'est une courte routine qui vérifie la validité du nombre entré avant de permettre au programme de continuer. Voici une courte routine servant à repérer une entrée erronée :

ROUTINE DE VÉRIFICATION DE L'ENTRÉE DU CHOIX

```
1 REM ROUTINE
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT « ENTREZ 1-9 »; CHOIX
40 IF CHOIX < 1 THEN LET L = 0
50 IF CHOIX >9 THEN LET L = 0
60 NEXT L
70 PRINT « VOTRE CHOIX ÉTAIT »; CHOIX
80 END
```

Plusieurs versions de BASIC permettent de simplifier cette routine en incluant un opérateur booléen dans le test, comme ceci :

```
10 LET L = 0
20 FOR L = 1 TO 1
30 INPUT « ENTREZ 1-9 »; CHOIX
40 IF CHOIX < 1 OU CHOIX >9 THEN LET L = 0
50 NEXT L
60 PRINT « VOTRE CHOIX ETAIT »; CHOIX
70 END
```

Ces routines illustrent une autre caractéristique de l'instruction INPUT. L'instruction INPUT demande au programme d'attendre une entrée en provenance du clavier. Le BASIC ne peut reconnaître le nombre entré que lorsque la touche RETURN est pressée. Vous devez donc vous souvenir de presser RETURN après avoir entré le nombre.

Mais le programme serait plus agréable s'il continuait dès qu'un nombre valide a été entré. Cela est possible en utilisant la fonction INKEY$. Le BASIC lit un caractère en provenance du clavier lorsqu'il rencontre le mot INKEY$. Le programme n'arrête pas, il continue jusqu'à la prochaine instruction du programme sans faire de pause. INKEY$ est donc généralement placé à l'intérieur d'une boucle. On pourrait utiliser la boucle suivante pour attendre la pression d'une touche : IF INKEY$ = « » THEN ... — en d'autres mots, si la touche pressée est « rien » (c'est-à-dire si aucune touche n'a été pressée), revenir et vérifier de nouveau. Voici une boucle correspondant à notre application :

```
FOR I = 1 TO 1
LET A$ = INKEY$
IF A$ = « » THEN LET I = 0
NEXT I
```

Le seul inconvénient de la fonction INKEY$ est qu'elle retourne un caractère de chaîne et non une valeur numérique. Lorsque nous désirons utiliser une structure ON ... GOTO, où l'on doit effectuer une sélection parmi plusieurs options (un branchement à conditions multiples), il est plus facile d'utiliser des nombres que des caractères. C'est ici que les fonctions VAL et NUM de BASIC sont utilisées. Elles convertissent des chaînes de caractères en nombres « réels » (en valeurs numériques, et non en codes ASCII représentant des nombres). Voici comment elles sont utilisées :

```
LET N = VAL(A$) ou LET N = NUM(A$)
```

Grâce aux fonctions NUM et VAL, le programme peut convertir en valeurs numériques les entrées obtenues avec INKEY$. Il n'est donc plus nécessaire d'appuyer sur la touche RETURN après avoir entré le nombre de sélection. Il est toutefois toujours préférable de vérifier la validité de l'entrée.

La portion de programme suivante comporte deux boucles imbriquées. La boucle interne attend l'entrée d'un caractère au clavier; la boucle externe convertit le caractère entré en une valeur numérique et vérifie sa validité :

```
FOR L = 1 TO 1
  PRINT « ENTREZ CHOIX (1-9) »
    FOR I = 1 TO 1
      LET A$ = INKEY$
```

```
    IF A$ = "" THEN LET I = 0
  NEXT I
LET CHOIX = VAL(A$)
  IF CHOIX < 1 THEN LET L = 0
  IF CHOIX >9 THEN LET L = 0
NEXT L
```

Voici finalement un programme BASIC complet pour le module * CHOIX *, comportant une entrée factice et des sous-programmes pour vérifier son bon fonctionnement. Rappelons de nouveau que les numéros de ligne ne servent qu'à effectuer la vérification, et que ceux-ci devront être modifiés lors de l'assemblage du programme final.

```
10 PRINT CHR$(12)
20 PRINT « CHOISIR L'UNE DES OPTIONS SUIVANTES »
30 PRINT
40 PRINT
50 PRINT
60 PRINT « 1. TROUVER ENREGISTREMENT (A PARTIR DU
   NOM) »
70 PRINT « 2. TROUVER ENREGISTREMENT (AVEC NOM
   PARTIEL) »
80 PRINT « 3. TROUVER ENREGISTREMENT (A PARTIR DE
   LA VILLE) »
90 PRINT « 4. TROUVER ENREGISTREMENT (A PARTIR DES
   INITIALES) »
100 PRINT « 5. LISTER TOUS LES ENREGISTREMENTS »
110 PRINT « 6. AJOUTER UN NOUVEL ENREGISTREMENT »
120 PRINT « 7. MODIFIER UN ENREGISTREMENT »
130 PRINT « 8. EFFACER UN ENREGISTREMENT »
140 PRINT « 9. QUITTER ET SAUVEGARDER »
150 PRINT
160 PRINT
170 LET L = 0
180 LET I = 0
190 FOR L = 1 TO 1
200 PRINT « ENTREZ CHOIX (1-9) »
210 FOR I = 1 TO 1
220 LET A$ = INKEY$
230 IF A$ = "" THEN LET I = 0
240 NEXT I
250 LET CHOIX = VAL(A$)
260 IF CHOIX < 1 THEN LET L = 0
270 IF CHOIX >9 THEN LET L = 0
280 NEXT L
290 ON CHOIX GOSUB 310, 330, 350, 370, 390, 410, 430,
   450, 470
300 END
310 PRINT « SOUS-PROGRAMME FACTICE 1 »
320 RETURN
330 PRINT « SOUS-PROGRAMME FACTICE 2 »
340 RETURN
350 PRINT « SOUS-PROGRAMME FACTICE 3 »
360 RETURN
370 PRINT « SOUS-PROGRAMME FACTICE 4 »
380 RETURN
390 PRINT « SOUS-PROGRAMME FACTICE 5 »
400 RETURN
410 PRINT « SOUS-PROGRAMME FACTICE 6 »
420 RETURN
430 PRINT « SOUS-PROGRAMME FACTICE 7 »
440 RETURN
450 PRINT « SOUS-PROGRAMME FACTICE 8 »
460 RETURN
470 PRINT « SOUS-PROGRAMME FACTICE 9 »
480 RETURN
```

La prochaine fois, nous étudierons les structures de fichier et commencerons à perfectionner la procédure INITIALISATION.

Variantes de basic

SPECTRUM

Dans le programme principal factice ainsi que dans tout le programme, remplacez PRINT CHR$(12) par CLS, et END par STOP.

ROUTINE DE VÉRIFICATION DE VALIDITÉ

```
 1 REM ROUTINE
 10 LET L = 0
 20 FOR L = 1 TO 1
 30 INPUT « ENTREZ 1-9 »; CHOIX
 40 IF CHOIX < 1 THEN LET L = 0
 50 IF CHOIX >9 THEN LET L = 0
 60 NEXT L
 70 PRINT « VOTRE CHOIX ÉTAIT »;
   CHOIX
 80 STOP
```

LISTING FINAL

```
 10 CLS
```

comme dans le texte principal jusqu'à

```
 240 NEXT I
 250 LET CHOIX = CODE A$ - 48
 260 IF CHOIX < 1 THEN LET L = 0
 270 IF CHOIX > 9 THEN LET L = 0
 280 NEXT L
 290 GOSUB (CHOIX*20 + 290)
 300 STOP
```

puis copier le programme principal de la ligne 310 à la ligne 480.

TAB

Certaines versions de l'Oric-1 ne répondent pas à la commande TAB, même si cette commande fait partie du BASIC de l'Oric-1; dans ce cas insérez cette ligne au début du programme :

```
5 LET S$ = "  "
```

Vous devez placer entre guillemets autant d'espaces qu'il y a de caractères affichés sur une ligne complète d'écran — 40 pour l'Oric-1. Puis lorsque le programme comporte l'instruction TAB(12), remplacez-la par LEFT$(S$,12), insérant le nombre de l'instruction TAB dans la fonction LEFT$( ).

CHR$(12)

Sur l'Oric-1, sur le Dragon 32, sur le Lynx et sur le BBC, remplacez PRINT CHR$(12) par CLS. Sur le Commodore 64 et sur le Vic-20, remplacez CHR$(12) par PRINT « touche shift + touche CLR/HOME »; un cœur inversé s'affichera. Consultez le manuel si cela ne vous paraît pas clair.

ON.. GOSUB

Cette instruction n'est pas disponible sur le Lynx, mais peut être remplacée par la ligne 290 du listing final du Spectrum.

VARIABLES

Voir « Variantes de basic » à la page 257.

INKEY$

Voir « Variantes de basic » à la page 175; les utilisateurs du Commodore 64 doivent remplacer LET A$ = INKEY$ par GET A$, et remplacer IF INKEY$ = "" THEN par : GET A$: IF A$ = " " THEN

Petites souris

Les concepteurs d'ordinateurs désirent abandonner le clavier en faveur d'un dispositif plus facile à utiliser, la « souris ».

Il n'y a pas très longtemps, il n'était possible d'accéder aux ordinateurs qu'au moyen de grosses machines à écrire nommées « télétypes ». Ces dispositifs étaient bruyants, encombrants et peu fiables; ils ont depuis été remplacés par l'ensemble unité d'affichage et clavier. L'unité d'affichage élimina de nombreux problèmes associés aux télétypes — dont la production de grandes quantités de bande de papier perforée lors de l'entrée des données. Cependant, le terminal mécanique et l'unité d'affichage-clavier sont limités par leur format caractère par caractère, ligne par ligne. L'utilisateur ne peut pas se déplacer rapidement sur l'écran (pour sélectionner des opérations sur un menu, pour modifier des données, ou pour transférer des fichiers et des programmes) sans devoir affronter les restrictions imposées par le curseur piloté au moyen du clavier. Le clavier offre plus de souplesse lorsqu'il est utilisé avec des terminaux graphiques ou dans des jeux électroniques.

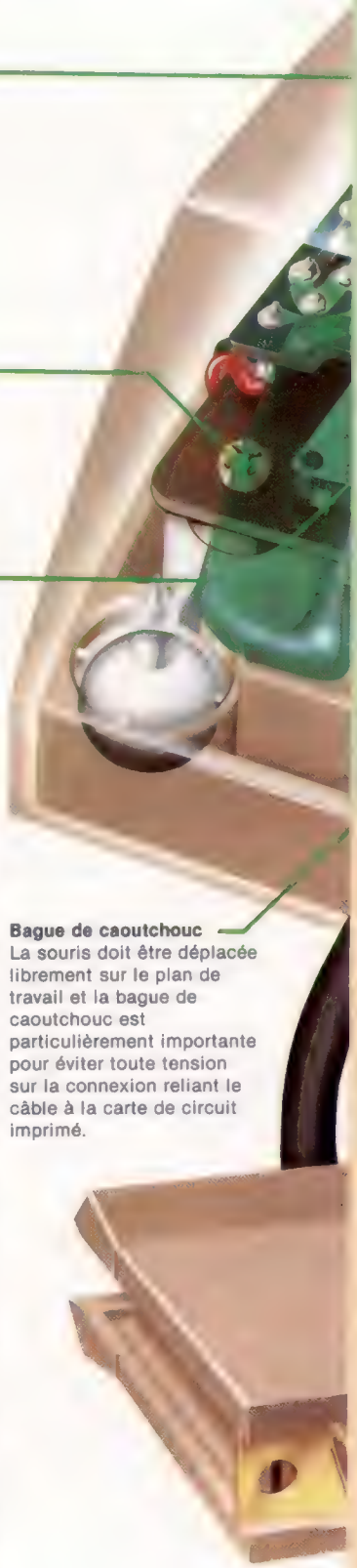
La plupart des ordinateurs domestiques offerts actuellement sur le marché sont munis de commandes de déplacement de curseur dans quatre directions. Elles permettent de placer le curseur dans un listing de programme ou dans un document à tout endroit où une modification doit être apportée. Mais le curseur ne peut être déplacé que caractère par caractère ou ligne par ligne. Si le curseur de texte pouvait être déplacé comme un curseur graphique, c'est-à-dire être manipulé librement au moyen d'une manette ou d'une boule roulante, les manipulations de données seraient évidemment beaucoup plus rapides.

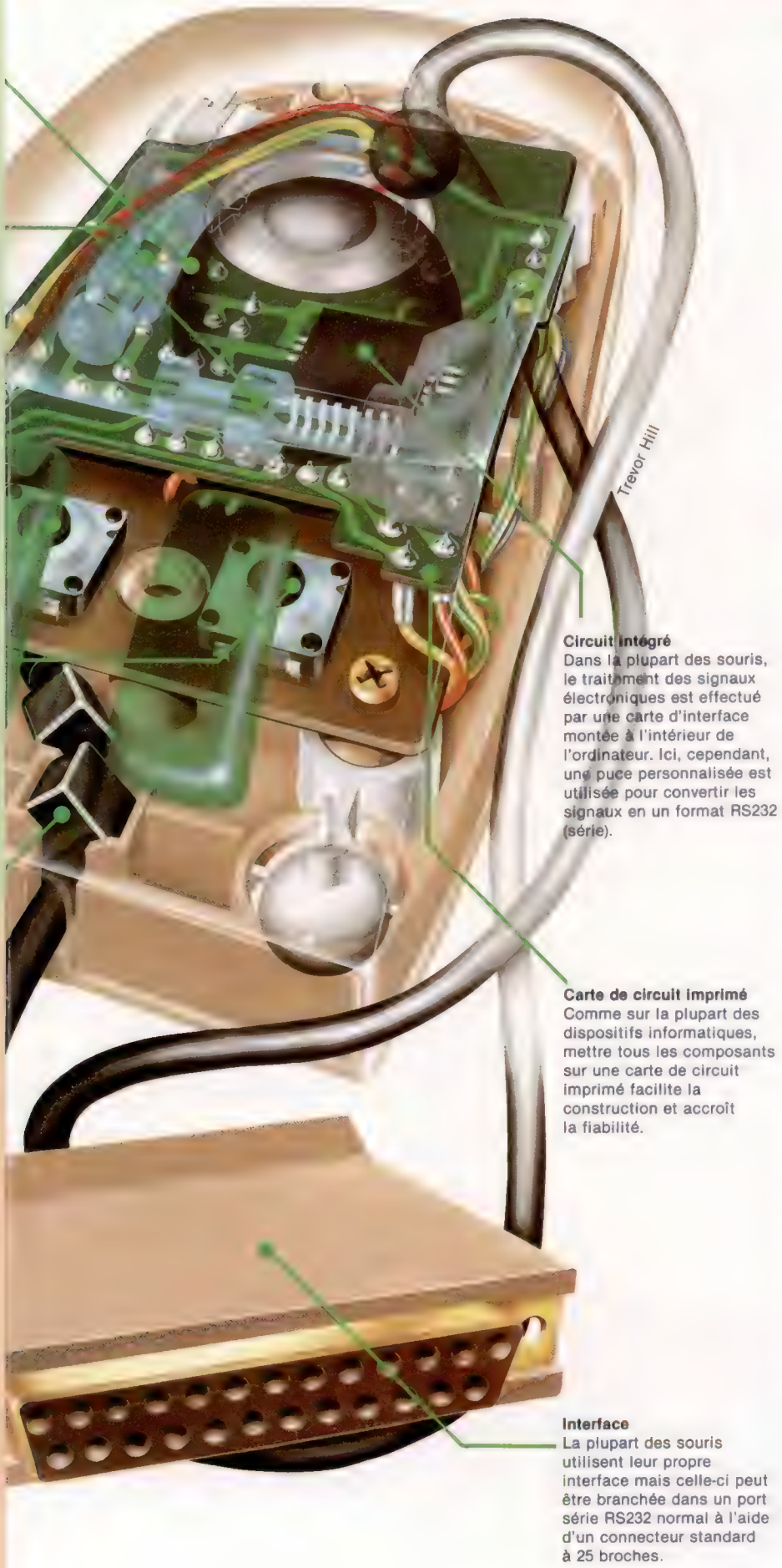
Trois souris aveugles
La plupart des nouveaux micro-ordinateurs d'affaires offrent, en configuration standard, une souris. Généralement, ces souris travaillent grâce à une boule située à l'intérieur de leur carrosserie.
(Cl. Ian McKinnell.)

Une solution à ce problème fut envisagée pour la première fois en 1960 au Stanford Research Institute en Californie; et la première « souris » — c'est ainsi qu'on nomma ce nouveau type de contrôleur d'écran — fut brevetée en 1970. Le dispositif fut nommé souris en raison de son apparence : une souris doit être assez petite pour être tenue dans le creux de la main, elle a une queue (le câble), et les premiers modèles avaient deux « oreilles » (les boutons de commande). Les boules roulantes et manettes conventionnelles ne sont pas utilisées parce que la précision qu'elles autorisent est superflue.

La souris fonctionne en détectant de façon assez précise la position qu'elle occupe sur toute surface plate et interprète tous les mouvements effectués dans les quatre directions. Ces mouvements sont directement convertis en mouvements du curseur — ou pointeur, comme il est souvent nommé — sur l'écran. Il existe deux méthodes principales pour générer les signaux. Dans les deux cas, une boule située en dessous de la souris roule sur la surface de déplacement du dispositif.

La rotation du roulement à bille est transférée à des axes cylindriques internes. Dans l'un des systèmes, les extrémités de ces cylindres sont munies de roues d'encodage qui possèdent des pistes alternées de matériaux conducteurs et non conducteurs. Les impulsions reçues sont comp-





Circuit intégré
Dans la plupart des souris, le traitement des signaux électroniques est effectué par une carte d'interface montée à l'intérieur de l'ordinateur. Ici, cependant, une puce personnalisée est utilisée pour convertir les signaux en un format RS232 (série).

Carte de circuit imprimé
Comme sur la plupart des dispositifs informatiques, mettre tous les composants sur une carte de circuit imprimé facilite la construction et accroît la fiabilité.

Interface
La plupart des souris utilisent leur propre interface mais celle-ci peut être branchée dans un port série RS232 normal à l'aide d'un connecteur standard à 25 broches.

tées par le système d'exploitation de la souris et lui permettent d'obtenir une lecture directe de la position du curseur à l'écran. Dans l'autre système, deux disques ajourés sont montés sur les axes. Une lumière est continuellement dirigée sur les disques et le faisceau est détecté optiquement de l'autre côté par une cellule photo-électrique. Les impulsions de lumière qui traversent les fentes de la roue sont converties en signaux électriques.

Il existe d'autres systèmes. Dans l'un de ceux-ci, la souris est utilisée avec une surface spéciale renfermant une configuration de points. Une lumière à l'intérieur de la souris illumine la zone de la surface parcourue et le trajet est détecté par une puce spéciale de traitement optique. Tout mouvement de la souris modifie la configuration détectée par la puce, ce qui permet à cette dernière de calculer exactement tous les mouvements effectués.

Dès que le curseur a été placé sur la position désirée de l'écran, on enregistre l'information dans l'ordinateur en pressant l'une des oreilles (boutons) de la souris. Le nombre de boutons varie d'un constructeur à l'autre. Certains systèmes en ont jusqu'à trois; Microsoft a choisi d'en utiliser deux. La souris du Lisa d'Apple n'en a qu'un. Les boutons peuvent aussi servir à sélectionner des éléments sur un menu ou permettre à la souris de commander le déplacement du curseur normal. Ces dispositifs peuvent être utilisés avec un logiciel très sophistiqué comme les programmes fournis sur le Lisa d'Apple. Ici le bouton est pressé une fois pour sélectionner une icône à l'écran et deux fois pour « ouvrir » l'application particulière concernée.

Le principal avantage des souris et de leur logiciel associé est qu'ils peuvent être utilisés très efficacement par des personnes ne sachant pas du tout dactylographier. Plutôt que d'avoir à taper le nom d'un programme ou à presser certaines lettres ou certains nombres pour sélectionner une fonction, l'utilisateur n'a qu'à placer le curseur sur l'application ou l'intervention désirée à l'aide de la souris, puis presser un bouton pour solliciter l'intervention.

Malheureusement, la souris n'élimine pas complètement l'utilisation du clavier (les nouvelles données doivent encore être entrées dans l'ordinateur), mais elle facilite grandement la manipulation de l'information. Des essais effectués par Apple pendant le développement du Lisa ont démontré qu'un utilisateur n'ayant jamais approché un ordinateur peut apprendre à utiliser le logiciel piloté par souris en moins de vingt minutes. Des logiciels comparables, exécutés sur des systèmes conventionnels, peuvent exiger jusqu'à vingt heures d'apprentissage, principalement par l'obligation de tout commander à partir du clavier et d'avoir à apprendre de nombreuses commandes compliquées. Les souris électroniques feront bientôt partie intégrante des ordinateurs domestiques. Elles sont efficaces et simples à utiliser et n'effraient pas les nouveaux venus à l'informatique comme un clavier traditionnel.

Travail de détective

Lorsque des données sont transmises d'un ordinateur à l'autre, il est toujours possible qu'elles soient altérées. Les codes de Hamming peuvent détecter et corriger ces erreurs.

Nous avons tous entendu parler d'énormes erreurs faites par des ordinateurs — comme poster 500 copies d'une même lettre publicitaire à une seule personne. La machine n'est généralement pas responsable de ce type d'erreur. Elle provient d'une inattention humaine, ou peut-être simplement d'une faute de frappe. L'ordinateur ne fait qu'amplifier le problème. Occasionnellement, des erreurs surviennent parce que le programme d'applications n'a pas été écrit pour parer à toute éventualité — comme l'envoi de factures produites par ordinateur au montant de 0,00 F.

Quelquefois les ordinateurs font des erreurs qui ne peuvent pas être attribuées à une intervention humaine. Celles-ci prennent généralement la forme d'« erreurs de bits ». Une erreur de bit survient lorsqu'un bit d'une section de données est inversé de 1 à 0 ou vice versa. Les erreurs de bits peuvent être causées par des défaillances du matériel, comme une puce RAM défectueuse. Voilà donc pourquoi de nombreux ordinateurs domestiques exécutent une routine de diagnostic d'erreur lors de la mise sous tension initiale.

La plupart des erreurs de bits sont cependant des « erreurs de logiciel » — des bits sont inversés même si toute la RAM a subi avec succès le test de diagnostic. Des erreurs de bits peuvent aussi survenir pendant des périodes de forte activité solaire, lorsque des particules atomiques pénètrent dans l'atmosphère et créent une interférence dans le fonctionnement électronique d'un micro-circuit. Dans certains domaines, des erreurs peuvent entraîner des conséquences désastreuses. Une variété de méthodes a donc été adoptée pour les détecter.

La plus simple est la vérification de parité. Le total de contrôle est une autre méthode, largement utilisée lors d'écriture de données sur une bande magnétique ou sur une disquette. Les données sont généralement manipulées par blocs de 128 octets, dont le dernier de ces octets à être lu ou écrit est un octet de total de contrôle. Cet octet représente la somme de tous les autres octets (chacun ayant une valeur comprise entre 0 et 255) modulo 256 — c'est-à-dire le reste de la division de la somme par 256. Voici un exemple :

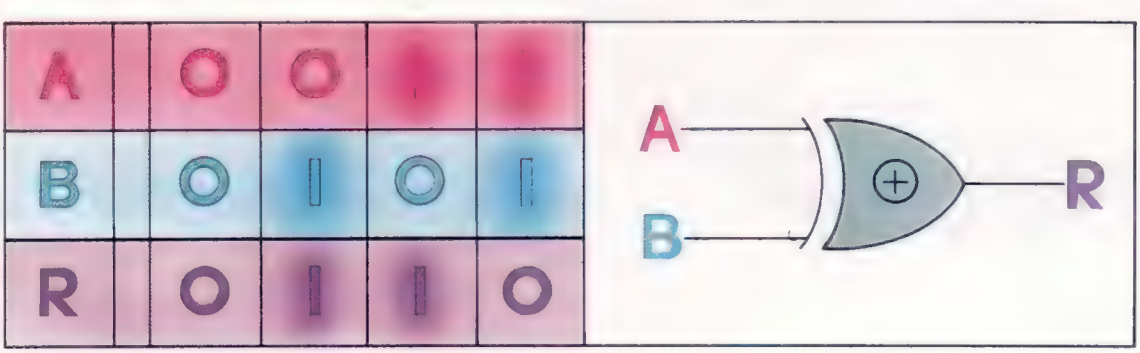
Données : 114, 67, 83... (121 autres valeurs)... 36, 154, 198
Total de ces 127 octets = 16 673
Total divisé par 256 = 65, reste 33
Donc total de contrôle = 33

Le total des octets (16 673) est égal à 65 lots de 256 plus un reste de 33 — la valeur qui est écrite dans le 128e octet à titre de total de contrôle. Lorsque l'ordinateur lit le bloc, il calcule son propre total de contrôle et si la valeur qu'il obtient diffère de 33, il sait qu'une erreur de bit est survenue pendant le processus d'enregistrement.

Avec la vérification de parité et le total de contrôle, l'ordinateur n'a aucun moyen de savoir quel bit de données a été inversé. Si l'erreur est survenue pendant une transmission, l'ordinateur-récepteur peut demander de retransmettre un octet particulier ou un bloc d'octets ; dans le cas d'une erreur d'enregistrement, il peut n'y avoir aucune manière de récupérer les données altérées.

Lorsque les erreurs sont tout à fait inacceptables, un système peut être utilisé pour à la fois les détecter et les corriger. Les codes de Hamming, appelés ainsi en l'honneur de leur inventeur R.W. Hamming, effectuent une opération.

Tout système de correction d'erreur est fondé sur le principe de la redondance. Les langages humains ont un certain degré de redondance — si une erreur de frappe survient dans un texte, ou si un bruit parasite fait perdre un mot ou deux dans une conversation téléphonique, il est



OU-exclusif
Une simple porte OU-exclusif a deux entrées et une sortie. Si les deux entrées sont à un niveau logique 0, la sortie est alors à un niveau logique 0. Si l'une des entrées est 1, la sortie est 1. Cependant, si les deux entrées sont 1, la sortie est 0. C'est cette dernière condition qui différencie la porte OU de la porte OU-exclusive. L'opération peut être représentée par une table de vérité. Lorsque la porte OU-exclusive a plus de deux entrées, la sortie sera 1 si l'entrée est constituée d'un nombre impair de 1. De tels dispositifs servent à effectuer les vérifications de parité et d'erreur de bits. (Cl. Kevin Jones.)

souvent possible de recréer les mots selon le contexte de la phrase.

Supposons que nous envoyons sur notre ordinateur un mot de x bits, composé de y bits de données réelles et de z bits de données redondantes ($x = y + z$). Dans notre explication de la parité, nous avions une valeur de 7 pour y et de 1 pour z. Pour les codes de Hamming, z devra être proportionnellement plus grand. Supposons maintenant qu'une erreur d'un bit survienne dans l'un des bits x (nos bits redondants z sont tout aussi exposés aux erreurs que les bits de données y). Si la probabilité d'erreur dans un mot est de 1 sur 1 million, la probabilité de deux erreurs dans un mot est de 1 sur 1 million de millions, nous ignorerons donc cette possibilité.

Lorsque les données sont reçues à l'autre extrémité, il y aura $x + 1$ possibilités. Soit il n'y aura aucune erreur, ou une erreur sur le premier bit, et ainsi de suite jusqu'au x^e bit. Avec les bits redondants z, nous pouvons représenter 2^z situations, donc l'expression suivante établit les conditions dans lesquelles le mot est à l'abri des erreurs d'un bit :

$$2^z > y + z + 1$$

Si y est égal à sept (pour les codes ASCII), z devra être quatre. Si y est quatre (comme dans l'exemple du tableau), z devra être trois. Cependant, si y est seize, z ne devra être augmenté qu'à cinq. On peut en conclure que les codes de Hamming sont plus efficaces pour des mots longs que pour des mots courts.

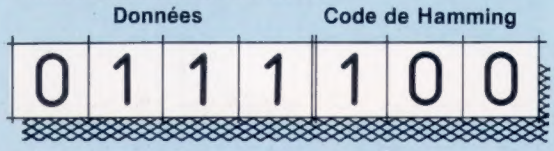
Dans un code de Hamming, chacun des bits redondants effectue une vérification de parité sur une combinaison différente de bits du mot. Si un bit est inversé pendant une transmission, un ou plusieurs bits de vérification seront faux et la combinaison de ces bits désignera le bit inversé du mot (voir l'exemple). L'ordinateur-récepteur n'a alors qu'à inverser à nouveau ce bit.

Les codes de Hamming fonctionnent en effectuant, avec chaque bit de Hamming, une vérification de parité sur diverses combinaisons de bits de données. Le nombre total de bits est divisé en plusieurs ensembles différents mais se chevauchant (définis de telle sorte que deux bits n'apparaissent jamais dans la même combinaison). L'ordinateur-récepteur effectue des vérifications de parité sur les mêmes ensembles que le dispositif émetteur utilise pour créer le code de Hamming. Si l'un des bits, y compris les bits de Hamming, a été inversé lors de la transmission, un ou plusieurs tests de parité seront négatifs. La combinaison de ces tests négatifs désignera un bit unique.

Certains ordinateurs utilisent même des codes de Hamming pour leurs opérations de mémoire interne. Dans un tel cas, il est possible d'enlever une puce RAM complète et de continuer à utiliser l'ordinateur ! Certains ordinateurs militaires exploitent le principe de redondance à l'extrême. Ils doublent chaque élément de l'ordinateur, et comparent les résultats des deux moitiés de l'appareil après chaque opération.

Comment fonctionne un code de Hamming

Supposons que nous désirions envoyer ces quatre bits de données.



A ceux-ci nous devrions ajouter un code de Hamming à 3 bits généré par l'ordinateur pour remplir les conditions suivantes :

En ne considérant que ces 4 bits, il doit y avoir un nombre pair de 1.

De façon similaire, parmi ces 4 bits, il doit y avoir un nombre pair de 1.

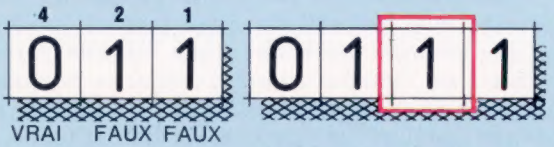
Et il doit y avoir un nombre pair de 1 dans cet ensemble également. Pour trouver les 3 bits qui satisfont à ces conditions, l'ordinateur doit résoudre trois équations simultanées.

Mais supposons que pendant la transmission, le troisième bit à partir de la gauche ait été altéré, c'est-à-dire inversé de 1 à 0.

Si l'ordinateur récepteur effectue le premier des trois tests sur les données, le test est négatif puisqu'il y a un nombre impair de 1. Il y a erreur, mais on ne peut pas encore savoir quel bit a été inversé.

De façon similaire, le deuxième test est négatif.

Cependant le troisième test est positif — il y a un nombre pair de 1.



C'est la combinaison des tests positifs et négatifs qui désigne l'endroit où se trouve le bit inversé. Si nous exprimons un test négatif par un 1 et un test positif par un 0, puis écrivons le résultat dans l'ordre inverse, nous obtenons le nombre binaire 3, ce qui nous indique que le troisième bit a été altéré et doit être inversé de 0 à 1.

Ce principe fonctionne toujours, même si l'un des bits de Hamming est inversé. Si les trois tests sont négatifs, par exemple, 111 indiquerait que le bit situé à l'extrême droite a été inversé, tandis que si les trois tests sont positifs, il n'y a aucune erreur. Ce type de code de correction d'erreur est limité.

Norbert Wiener

L'enfant prodige dont les travaux ont entraîné la naissance de la cybernétique.

Norbert Wiener est né en 1894 au Missouri (États-Unis). Diplômé en mathématiques à l'âge de quatorze ans, il reçut un doctorat à l'âge de dix-huit ans. Il étudia ensuite avec David Hilbert à Göttingen, en Allemagne.

Ce n'est qu'à la fin de sa vie que Wiener apporta une contribution à l'informatique. Pendant de nombreuses années il travailla au Massachusetts Institute of Technology, étudiant la physique probabiliste, et se concentrant sur l'étude statistique du mouvement des particules dans un liquide (le mouvement brownien). Les mouvements des particules étaient si imprévisibles qu'il était impossible de les décrire à l'aide des notions de physique déterministes traditionnelles. Il était préférable d'appliquer une méthode « probabiliste » qui pouvait prédire l'emplacement probable d'une certaine particule à un moment donné. Lorsque la Seconde Guerre éclata, il offrit ses services au gouvernement américain et commença à étudier les problèmes mathématiques impliqués dans les techniques de visée d'un objectif en mouvement. Le développement de systèmes de guidage de tir automatique combiné avec ses études en physique probabiliste et l'intérêt qu'il manifestait pour divers sujets allant de la philosophie à la neurologie le conduisirent à publier en 1948 un livre intitulé *Cybernétique*.

La cybernétique est la science constituée par l'ensemble des théories relatives aux communications et à la régulation dans l'être vivant ou dans la machine. Wiener vit dans l'information une quantité tout aussi importante que l'énergie ou la matière : un fil de cuivre, par exemple, pouvait être étudié pour l'énergie qu'il pouvait transmettre ou pour l'information qu'il pouvait communiquer. La révolution promise par l'ordinateur est fondée partiellement sur cette constatation : la puissance sera de moins en moins due au fait de posséder des terres, des industries ou des commerces, mais en contrôlant l'information. Sa contribution à la science informatique ne fut pas la création d'un produit mais la mise en place d'un environnement intellectuel dans lequel il devint possible de créer des ordinateurs et des automates.

Le mot cybernétique est d'origine grecque et signifie « science du gouvernement ». Wiener étudia la régulation de la machine à vapeur de James Watt qui définissait automatiquement la vitesse de la machine. Il se rendit compte que pour développer les ordinateurs, il était nécessaire de leur faire imiter les êtres humains en ce qu'ils gouvernent leurs propres activités.

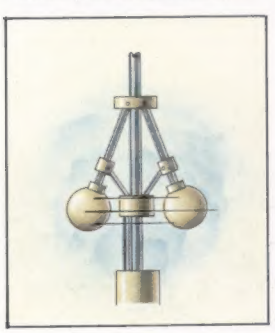
Le thermostat domestique est un exemple de système de contrôle. Il régularise la chaleur en fonction des fluctuations de la température au-dessus ou au-dessous d'un niveau optimal. Seul le réglage de ce niveau doit être effectué par un être humain. Wiener nomme cette faculté d'autorégulation et de contrôle le « feedback négatif » — « feedback » parce que la sortie du système (la chaleur) modifie le comportement futur du système et « négatif » parce que les changements commandés par le thermostat servent à restaurer le niveau de température réglé. Un système qui peut faire la même chose et également choisir sa propre température (et d'autres objectifs) est dit « système à feedback positif ». Lorsqu'un automate peut agir de la sorte et plusieurs fois, il approche la condition humaine.

La théorie cybernétique de Wiener peut être considérée comme une « super-science » (une science des sciences). Elle a encouragé la poursuite de recherches pour les systèmes de contrôle et de gestion de l'information.

L'information peut être étudiée d'une façon statistique, indépendamment de toute signification. Par exemple, en examinant la fréquence d'utilisation de certains symboles, il est possible de percer plusieurs types de codes. En français, la lettre « e » est la plus utilisée, et la lettre « t » est celle dont la fréquence vient au deuxième rang. En analysant de larges échantillons de texte français, on peut identifier les lettres clés et donc de commencer à déchiffrer le code.

Wiener est mort en 1964, avant le début de la révolution micro-informatique. Il avait pourtant prévu et commenté de nombreux problèmes liés à cette nouvelle technologie.

Régulateur de vitesse
Wiener était fasciné par le concept du régulateur de la machine à vapeur, l'un des meilleurs et des plus simples exemples de « feedback négatif ». Deux poids sont reliés au moyen de bras articulés à l'axe de rotation qui est lui-même relié au volant de la machine à vapeur. Plus la vitesse de la machine augmente, plus les poids s'écartent de l'axe. Ce mouvement, grâce à une liaison adéquate, ferme légèrement le registre de vapeur de la machine. Cela a pour effet de stabiliser la vitesse de la machine à tout niveau défini par l'opérateur. Des ordinateurs peuvent gérer des contrôles beaucoup plus évolués, mais le principe demeure le même. (Cl. Kevin Jones.)



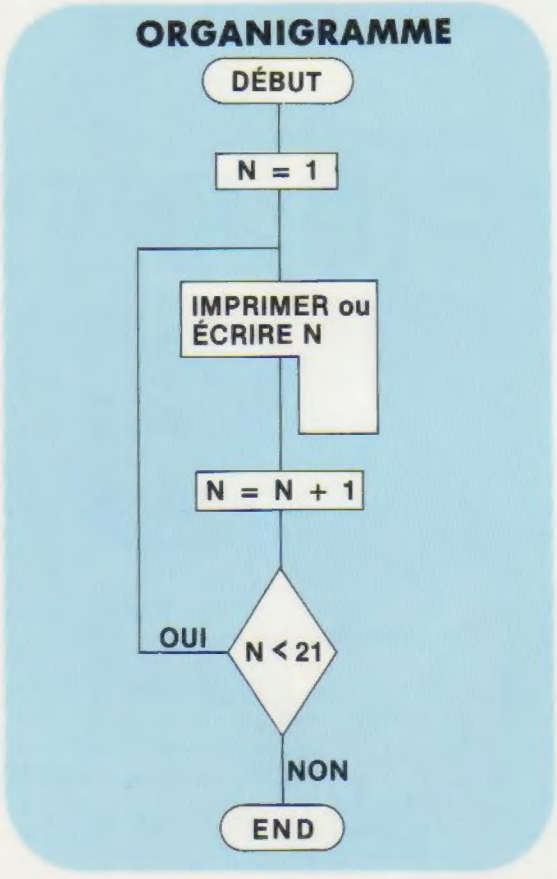
PROGRAMME N° 5

Nous avons introduit la notion de boucle et l'instruction GOTO dans notre programme précédent. Mais, aujourd'hui, nous voulons contrôler la longueur et la durée d'une boucle.
La solution est l'instruction IF. Si la condition est remplie, la machine saute l'instruction GOTO et ensuite l'instruction de la ligne suivante. S'il n'y a pas de ligne suivante le programme s'arrête.

Voici le programme qui compte de 1 à 20 puis stoppe.

```
10 HOME
20 N = 1
30 PRINT N
40 N = N + 1
50 IF N < 21 THEN GOTO 30
```

L'instruction IF opère de la manière suivante :
IF impression arithmétique THEN une instruction quelconque.
L'expression arithmétique est d'abord évaluée. Si elle donne comme résultat 0 (FAUX), tout le reste de la ligne de programme est ignoré et la machine exécute l'instruction de la ligne suivante. Si l'expression arithmétique n'est pas 0 (VRAI), la machine exécute la suite de l'instruction.



AUTRES TYPES DE BOUCLES

Instruction FOR NEXT.

Qu'elles soient exécutées par des programmes ou des aéroplanes, les boucles comportent toujours un haut et un bas.

Dans le programme précédent, la ligne 30 représente le haut et la ligne 50 le bas de la boucle. Le programme imprime des nombres entiers de 0 à 20 inclus. Le nombre 20 représente la limite de la boucle.

Une seconde manière de regarder une boucle consiste à introduire une instruction que nous n'avons pas encore eu l'occasion de donner, l'instruction FOR.

Nous allons l'introduire dans le programme précédent.

```
§110 HOME
§120 FOR I = 1 TO 20
§130 ?I
140  NEXT I
```

Taper RUN 110 afin d'exécuter le programme.
Si vous tapez RUN, le système exécute le programme à la ligne 10 (le plus petit numéro de la ligne).
La ligne 120 contient la nouvelle instruction FOR. Elle commence par attribuer la valeur 0 à N, ce

qui correspond à la ligne 20. Le système exécute ensuite la ligne 130. Le numéro 140 correspond au bas de la boucle.

La variable N augmente d'une unité avant d'être comparée à la limite supérieure fixée par l'instruction FOR. Tant que la variable N ne dépasse pas cette limite, la machine passe à l'instruction suivante. Si N dépasse la limite, le programme quitte la boucle pour exécuter l'instruction suivant le NEXT. Dans le cas présent, le programme sort de la boucle et s'interrompt puisqu'il ne rencontre plus d'instructions.

L'instruction FOR/NEXT vous épargne une instruction lorsque vous formez les boucles et facilite la rédaction de ce genre de programme.

Cette méthode facilite également la lecture des instructions. Vous ne devez plus vérifier qu'une seule instruction au lieu de trois. Pour connaître le déroulement de la boucle, il suffit de chercher un NEXT possédant la même variable que le FOR pour connaître le bas d'une boucle FOR/NEXT.

Si vous voulez imprimer (PRINT) uniquement les nombres pairs de 0 à 20, vous pouvez utiliser le programme.

```
10 HOME
20 N = 0
25 PRINT N
30 N = N + 2
40 IF N < = 20 THEN GOTO 25
```

Dans la ligne 40, où nous avons ajouté 2 à la variable N, la boucle avance de 2. Pour avancer de 2 dans une boucle FOR, taper :

```
FOR N = 0 TO 20 STEP 2
```

```
10 HOME
20 FOR I = 0 TO 20 STEP 2
30 PRINT I
35 NEXT I
40 END
```

La valeur de l'avancement (STEP) peut être tout nombre entier acceptable par la machine. On peut avoir aussi un avancement arrière comme ceci.

```
5  HOME
10 FOR I = 20 TO 0 STEP - 2
20 PRINT I
40 NEXT I
50 END
```

Les applications de l'instruction FOR ne sont pas illimitées. Les boucles FOR/NEXT peuvent être emboîtées, par exemple, mais non croisées. Voici un exemple de boucles emboîtées et un exemple à ne pas suivre. (Les boucles se chevauchent.)

```
5  HOME
10 HOME
20 FOR I = 1 TO 5
30 INPUT « NUMERO DE L'ELEVE : »; N

40 FOR J = 1 TO 3
50 INPUT « NOTE DE L'ELEVE : »; E

60 NEXT J
70 NEXT I
```

Les boucles sont emboîtées
EXÉCUTION POSSIBLE

```
$RUN
NUMERO DE L'ELEVE : 1
NOTE DE L'ELEVE   : 12
NOTE DE L'ELEVE   : 14
NOTE DE L'ELEVE   : 9
NUMERO DE L'ELEVE : 2
NOTE DE L'ELEVE   : 14
NOTE DE L'ELEVE   : 16
NOTE DE L'ELEVE   : 8
NUMERO DE L'ELEVE : 3
NOTE DE L'ELEVE   : 13
NOTE DE L'ELEVE   : 11
NOTE DE L'ELEVE   : 7
NUMERO DE L'ELEVE : 4
NOTE DE L'ELEVE   : 5
NOTE DE L'ELEVE   : 8
NOTE DE L'ELEVE   : 14
NUMERO DE L'ELEVE : 5
NOTE DE L'ELEVE   : 18
NOTE DE L'ELEVE   : 19
NOTE DE L'ELEVE   : 20
```

```
$LIST
5  HOME
10 HOME
20 FOR I = 1 TO 5
30 INPUT « NUMERO DE L'ELEVE : »; N

40 FOR J = 1 TO 3
50 INPUT « NOTE DE L'ELEVE : »; E

60 NEXT I
70 NEXT J
```

Les boucles se chevauchent
EXÉCUTION IMPOSSIBLE